Logarithmic Opinion Pools for Conditional Random Fields

Andrew Smith

Doctor of Philosophy

Institute for Communicating and Collaborative Systems

School of Informatics

University of Edinburgh

2007

Abstract

Since their recent introduction, conditional random fields (CRFs) have been successfully applied to a multitude of structured labelling tasks in many different domains. Examples include natural language processing (NLP), bioinformatics and computer vision. Within NLP itself we have seen many different application areas, like named entity recognition, shallow parsing, information extraction from research papers and language modelling. Most of this work has demonstrated the need, directly or indirectly, to employ some form of regularisation when applying CRFs in order to overcome the tendency for these models to overfit. To date a popular method for regularising CRFs has been to fit a Gaussian prior distribution over the model parameters.

In this thesis we explore other methods of CRF regularisation, investigating their properties and comparing their effectiveness. We apply our ideas to sequence labelling problems in NLP, specifically part-of-speech tagging and named entity recognition.

We start with an analysis of conventional approaches to CRF regularisation, and investigate possible extensions to such approaches. In particular, we consider choices of prior distribution other than the Gaussian, including the Laplacian and Hyperbolic; we look at the effect of regularising different features separately, to differing degrees, and explore how we may define an appropriate level of regularisation for each feature; we investigate the effect of allowing the mean of a prior distribution to take on non-zero values; and we look at the impact of relaxing the feature expectation constraints satisfied by a standard CRF, leading to a modified CRF model we call the *inequality CRF*. Our analysis leads to the general conclusion that although there is some capacity for improvement of conventional regularisation through modification and extension, this is quite limited. Conventional regularisation with a prior is in general hampered by the need to fit a hyperparameter or set of hyperparameters, which can be an expensive process.

We then approach the CRF overfitting problem from a different perspective. Specifically, we introduce a form of CRF ensemble called a *logarithmic opinion pool* (LOP), where CRF distributions are combined under a weighted product. We show how a LOP has theoretical properties which provide a framework for designing new overfitting reduction schemes in terms of diverse models, and demonstrate how such diverse models may be constructed in a number of different ways. Specifically, we show that by constructing CRF models from manually crafted partitions of a feature set and combining them with equal weight under a LOP, we may obtain an ensemble that significantly

outperforms a standard CRF trained on the entire feature set, and is competitive in performance to a standard CRF regularised with a Gaussian prior. The great advantage of LOP approach is that, unlike the Gaussian prior method, it does not require us to search a hyperparameter space.

Having demonstrated the success of LOPs in the simple case, we then move on to consider more complex uses of the framework. In particular, we investigate whether it is possible to further improve the LOP ensemble by allowing parameters in different models to interact during training in such a way that diversity between the models is encouraged.

Lastly, we show how the LOP approach may be used as a remedy for a problem that standard CRFs can sometimes suffer. In certain situations, negative effects may be introduced to a CRF by the inclusion of highly discriminative features. An example of this is provided by gazetteer features, which encode a word's presence in a gazetteer. We show how LOPs may be used to reduce these negative effects, and so provide some insight into how gazetteer features may be more effectively handled in CRFs, and log-linear models in general.

Acknowledgements

I would like to thank my supervisor Dr. Miles Osborne for the many useful comments, interesting ideas, immense enthusiasm and general support he provided throughout the period of study leading to this thesis. I would also like to thank colleagues at the University of Edinburgh for many stimulating and fruitful discussions. These include, but are not restricted to, Trevor Cohn, Markus Becker, David Talbot, Stephen Clark, James Curran and Mirella Lapata.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Andrew Smith*)

Preface

Parts of the work presented in this thesis have been published at a number of language conferences. The content of each conference paper corresponds, roughly, to a chapter in the thesis. The chapters in question are:

(1) Chapter 5, which corresponds to Smith and Osborne (2005).

(2) Chapter 6, which corresponds to Smith et al. (2005).

(3) Chapter 8, which corresponds to Smith and Osborne (2006).

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Conditional Random Fields (CRFs) were introduced in 2001 by Lafferty et al. (2001) and currently represent a state-of-the-art approach to structured labelling problems in natural language processing (NLP). Although originally employed for sequence labelling tasks such as noun-phrase chunking (NPC) (Sha and Pereira, 2003) and named entity recognition (NER) (McCallum and Li, 2003), CRFs have recently been applied to problems that involve a more complex structure of dependencies between the objects being labelled. Examples include extraction of information from research papers (Peng and McCallum, 2004), semantic role labelling (Cohn and Blunsom, 2005) and parsing (Clark and Curran, 2004). The widespread acceptance of the CRF as the model of choice for many labelling problems within the NLP community is evidenced by the fact that its use has spread beyond English to many other languages, including Hindi (Li and McCallum, 2003), Chinese (Peng et al., 2004) and Japanese (Kudo et al., 2004). CRFs have also seen increasing application in domains other than NLP, such as computer vision (He et al., 2004; Wang and Ji, 2005) and computational biology (Culotta et al., 2005).

CRFs were originally introduced to overcome some of the weaknesses of related sequence labelling models. Hidden Markov Models (HMMs) (Rabiner, 1989), for example, had enjoyed widespread success on a number of labelling tasks in NLP, including part-of-speech tagging (Kupiec, 1992), information extraction (Freitag and McCallum, 2000) and shallow parsing (Molina and Pla, 2002). However, HMMs are *generative* models which model a joint distribution over both observations and their labels. This joint structure means that HMMs typically enforce strict conditional independence assumptions between elements of the observations. Consequently, it is difficult to tractably encode arbitrary dependencies between observation elements in an HMM, so

the modelling power is restricted. *Discriminative* approaches, by contrast, model only the distribution of the labels *given* the observations. Examples of discriminative models include sequential maximum entropy (ME) models (Ratnaparkhi, 1996), of which a special case is the maximum entropy Markov model (MEMM) (McCallum et al., 2000). The discriminative nature of these models allows the specification of arbitrary, non-independent properties of the observations via a set of *features*. For some tasks, this extra facility means that important dependencies between the observations, which would be hard to tractably encode in an HMM, can easily be included in the model. This often leads to increased performance (McCallum et al., 2000). Conventional discriminative models for sequence labelling typically specify a label distribution which is normalised at each point in the sequence. As a result of this structure, these models suffer a limitation which has become known as *label bias* (Bottou, 1991), and relates to an undesired biasing effect in the label distribution towards certain labellings. CRFs are also discriminative models, but in contrast to conventional discriminative models, they specify a label distribution which is *globally* normalised over the whole sequence. In doing so they avoid this bias effect. CRFs can therefore be seen as a solution to the weaknesses of both generative sequencing models and conventional discriminative sequencing models, taking advantage of increased modelling power over HMMs while simultaneously avoiding the bias effects of MEMMs.

Despite the clear advantages CRFs possess over alternative models, they do have certain shortcomings. At a high level these deficiencies may be divided into two categories: *scaling* and *overfitting*. The first of these, *scaling*, relates to both the computational and storage demands of CRF training, and how these scale with the complexity of the task and the size of the label set. In general, CRFs take longer to train than comparable discriminative models, and usually take considerably longer than HMMs. With all these types of model, the likelihood function is often used as an objective for model training. In a supervised training setting, with HMMs the likelihood function decouples into separate functions consisting of disjoint sets of parameters. These separate functions may be maximised independently and the maximum likelihood solution with respect to the model parameters can be found analytically. With discriminative models, however, the story is a little different because the likelihood function usually includes a normalising function which couples the model parameters, generally preventing a maximum likelihood solution in a closed form. As a result, solving for the model parameters usually demands the use of an iterative numerical optimiser. This is the reason that training time for discriminative models, like CRFs and MEMMs, is

generally longer than that for generative models, like HMMs. In addition, evaluation of the normalising function for a CRF is typically computationally more demanding than that for other discriminative models because the normalisation in a CRF is global. Similar remarks hold for the high memory demands of CRF training. In general, efficient evaluation of the normalising function for a CRF requires caching a large amount of information. For tasks that involve complex dependencies between the objects being labelled and/or a large label set, these heavy storage demands make CRF training impossible without an explicit strategy to overcome them. Cohn (2006) describes two approaches to overcome these scaling problems when applying CRFs to larger tasks. In this thesis we do not address these issues directly but instead focus on the second CRF shortcoming mentioned above, namely *overfitting*.

Many of the important dependencies in a label distribution may be modelled effectively by a CRF with a large number of highly expressive features. However, this powerful modelling capability comes hand-in-hand with the increased danger that some degrees of freedom contained in the model may fit to the idiosyncrasies of the training data itself, rather than the systematic properties of the underlying distribution. CRFs therefore exhibit a tendency to overfit the training data to a greater degree than alternative models. Indeed, most work with CRFs to date has demonstrated the need for some form of regularisation to be applied.

Conventional approaches to regularising log-linear models in general, and CRFs in particular, have focused on the use of a Gaussian prior over the model parameters (Chen and Rosenfeld, 1999; Sha and Pereira, 2003). This approach has been shown to be effective as a regularising strategy for a number of different tasks. However, despite its popularity there is no tractable way to determine the optimal hyperparameters of a prior distribution such as a Gaussian. In most cases fitting a Gaussian involves an element of trial-and-error, and is largely seen as a "black art". It is therefore desirable to investigate other approaches to CRF regularisation, either by improving and refining existing methods or by formulating an alternative paradigm. In this thesis we address these issues by introducing a new framework for CRF regularisation that does not require any hyperparameter search. The model is called a *logarithmic opinion pool*.

1.1 Contributions of the Thesis

The main contributions of this thesis fall into three categories:

1. **Analysis of Conventional CRF Regularisation**. As alluded to above, the current approach to CRF regularisation revolves around the use of a Gaussian prior distribution over the model parameters. Typically, for simplicity, certain assumptions are usually made when employing this distribution, such as constraining the Gaussian variance to be fixed across all parameters and forcing the Gaussian mean to be zero for all parameters. In this thesis we venture beyond this basic scenario to consider a range of other possibilities for conventional regularisation of CRFs. Specifically, we consider choices of prior distribution other than the Gaussian; we look at the effect of regularising different features separately, to differing degrees, and explore how we may define an appropriate level of regularisation for each feature; we investigate the effect of allowing the mean of a prior distribution to take on non-zero values; and we look the impact of relaxing the feature expectation constraints satisfied by a standard CRF, leading to a modified CRF model we call the *inequality CRF*. Our general conclusion is that although there is some capacity for improvement of conventional regularisation through modification and extension, this is quite limited. Conventional regularisation with a prior is in general hampered by the need to fit a hyperparameter or set of hyperparameters. For a large number of hyperparameters the hyperparameter search space quickly becomes too large to explore efficiently.

2. **New Framework for CRF Regularisation**. In this thesis we introduce a new framework for CRF regularisation called a *logarithmic opinion pool (LOP)*. The LOP combines a set of CRFs in a weighted product. The great advantage of the LOP approach in comparison with a conventional prior is that the LOP avoids the need to fit a hyperparameter or set of hyperparameters. For this reason, we refer to the LOP approach as "parameter-free". This aspect makes the LOP cleaner, and easy to implement and run. We show that the success of a LOP is determined by the *diversity* of the models from which is composed, and we investigate how this diversity may be introduced and adjusted. Overall, we show that a LOP, with an appropriate choice of CRFs in the weighted product, may obtain results that are as good as, or better, than those obtained through conventional regularisation.

3. **Applications of the Framework**. Having introduced the LOP, we go on to

show how it may be used as a remedy for a much under-appreciated problem with CRFs: in some circumstances, certain negative effects may be introduced to a CRF by the inclusion of highly discriminative features. An example of this is provided by gazetteer features, which encode a word's presence in a gazetteer. We demonstrate and explain the negative effect of such features, and show how a LOP may be used to reduce its impact. In doing so we provide some insight into how gazetteer features may be more effectively handled in CRFs, and log-linear models in general.

1.2 Structure of the Thesis

Chapter 2 provides a background to the theory behind CRFs and covers the relevant literature. We look in more detail at the relationship between CRFs and previous, related models and examine more closely the advantages that CRFs possess. We also describe the structure of a CRF from a graphical model point of view, and look at the algorithms for CRF training and decoding. Also in this chapter we briefly cover previous work on the two tasks we use for our experiments.

In Chapter 3 we present the experimental setup. We describe the two labelling tasks, named entity recognition and POS tagging, that we use to compare the models. We also discuss the experimental pipeline and software architecture employed. Lastly, we look at the measures used to evaluate and compare the performance of the different models.

Chapter 4 presents what we call the *reference models*. These are standard CRF models which act as the base models for our experimental comparisons and are built upon and extended in later chapters. Also in this chapter we demonstrate empirically the tendency for a CRF to overfit a training dataset of small size.

In Chapter 5 we investigate conventional regularisation for CRFs and propose some extensions to existing methods. We start by looking at the simplest approach, *feature cutoff*, and then move on to investigate the use of a prior distribution. We compare the standard Gaussian prior to two other distributions, the Laplacian and the Hyperbolic. We also explore how the level of regularisation applied to each model parameter can be made to vary across parameters, and investigate several ways in which this level may be set. Finally, we introduce an alternative model for conventional regularisation, called the *inequality CRF*, and show how this is related to existing approaches.

In Chapter 6 we introduce an alternative paradigm for CRF regularisation based on

a *logarithmic opinion pool* (LOP). We describe the theory behind LOPs and investigate some of their properties. We show that the performance of a LOP depends on how diverse its constituent models are, and explore different ways in which this diversity may be introduced to the model.

In Chapter 7 we expand on Chapter 6 by investigating how to introduce diversity into a LOP through *co-operative training* of the constituent models. Here, rather than using independently pre-trained constituent models in the LOP, we train the parameters in all constituent models together simultaneously, allowing interaction between them. We describe the framework for co-operative training and discuss what constitutes a suitable objective function. We present results to show how a co-operatively trained LOP compares to one with independently pre-trained constituent models.

In Chapter 8 we investigate how certain negative effects may be introduced to a CRF by the inclusion of highly discriminative features. We explore the nature of these negative effects and look at how they may be overcome. Specifically, we show how the LOP approach of the previous chapters may be used as a solution to the problem, and therefore how this represents an application of the ideas in the thesis.

Finally, in Chapter 9, we summarise our findings from the previous chapters and conclude the thesis.

Chapter 2

Background

In this chapter we provide some background for the rest of the thesis. The chapter is intended to cover the main concepts required to understand later chapters, and to examine the relevant literature in the field. We start, in section 2.1, by situating CRFs alongside previous, comparable sequence labelling models. Specifically, we look at the strengths and weaknesses of generative sequence models, such as HMMs, and see how these models were superseded by discriminative sequence models such as MEMMs. We also briefly look at the shortcomings of these discriminative models, such as the *label bias* problem. This naturally leads to the need for a model which corrects for these weaknesses, and in section 2.2 we introduce the CRF as such a model. We define the CRF, and show its structure as a graphical model, in section 2.2.1. We then give training and decoding algorithms for CRFs in sections 2.2.2 and 2.2.3 respectively. Section 2.2.4 describes applications of CRFs to date, both in NLP and in other domains. Finally, in section 2.3, we move on to consider the phenomenon of overfitting, which is particularly acute in CRFs. We first explore the concept in general, then, in section 2.3.1, we look at specific examples of schemes used to reduce overfitting, for CRFs and other log-linear models.

2.1 Generative Models to Discriminative Models

When CRFs (Lafferty et al., 2001) were first introduced, they were designed, in part at least, to address some of the weaknesses of previous probabilistic models used for sequence labelling. In this section we consider the properties of these earlier models, and see how their shortcomings led to the development of the CRF. We divide the models into two broad categories: **generative** and **discriminative**.

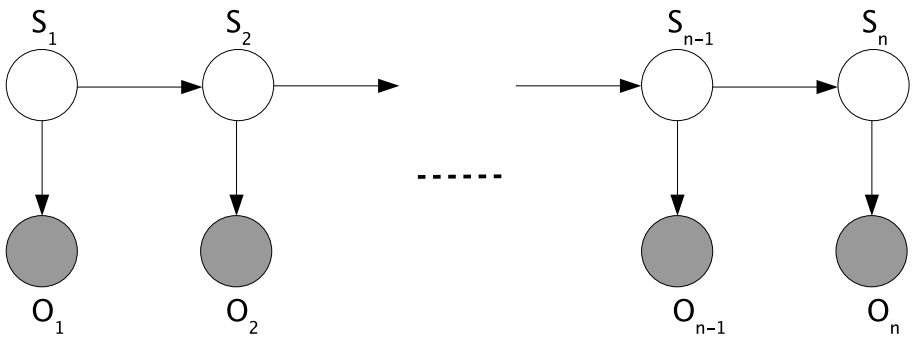

Figure 2.1: HMM graphical model structure.

2.1.1 Generative Sequence Models

Generative models assign a joint probability to a paired observation and label sequence. Examples of such models include **hidden Markov models (HMMs)** and **stochastic grammars (SGs)**. Hidden Markov models have been applied to a number of labelling tasks in speech recognition (Rabiner, 1989) and natural language processing, including part-of-speech tagging (Kupiec, 1992), information extraction (Freitag and McCallum, 2000) and shallow parsing (Molina and Pla, 2002). They have also been applied to similar tasks in other domains. For example, biological applications include the modelling of secondary structure in protein families and intron splicing in eukaryotic genes (Balay et al., 2001).

Figure 2.1 shows an HMM as a graphical model. The model consists of a set of hidden states, or labels[1], represented by random variables S_t, and a set of observable elements represented by random variables O_t. In the figure the observable variables are shaded, a convention we will use for graphical models throughout this chapter. In most tasks in NLP both the state variables and the observation variables are discrete. For example, in a part-of-speech tagging task[2] the state variables S_t could represent POS tags while the observation variables O_t represent words.

From the graph we can see that certain conditional independence relationships exist between the states and the observations. Specifically, S_{t+1} is conditionally independent of S_{t-1} given S_t. More generally, conditioning on S_t renders S_u and S_v independent for all $u < t$ and $t < v$. In addition, O_t is independent of all other variables given S_t. However, conditioning on an observation does not yield any conditional independencies, so

[1] In the thesis we assume there is a one-to-one mapping between states and labels, and so we treat the two terms as interchangeable. In general though, this need not be the case.

[2] We explain POS tagging in Chapter 3.

the "future is independent of the past given the present" rule only holds if by "present" we are referring to the current state.

Being a generative model, the HMM assigns a joint probability to an observation and label sequence. This joint distribution may be factorised according to the conditional independence relationships above. The distribution then becomes:

$$p(\mathbf{s},\mathbf{o}) = p(s_1)\prod_{t=1}^{T-1} p(s_{t+1}\,|\,s_t)\prod_{t=1}^{T} p(o_t\,|\,s_t) \tag{2.1}$$

where t runs over the T positions (words) in the sequence. The HMM is therefore parameterised by a set of local conditional probability distributions $p(s_{t+1}\,|\,s_t)$ and $p(o_t\,|\,s_t)$. In most cases we assume the model is *stationary*, meaning that the parameters in the distributions are independent of t. For a given HMM structure, the value of the parameters may be estimated by maximising an objective function over the training data. The simplest objective function is the **joint likelihood**, although other objectives may be used. If the training data consists of fully labelled data, with both observation sequences and state sequences given, the parameters may be found easily using the ratio of the relevant event counts. If the data is only partially labelled, or unlabelled, the parameters may be be found using an iterative algorithm such as Expectation-Maximisation (EM).

Once an HMM has been trained it may be used to label, or *decode*, previously unseen observation sequences. The task is to find the optimal state sequence $\mathbf{s}^*$ with:

$$\mathbf{s}^* = \operatorname*{argmax}_{\mathbf{s}} p(\mathbf{s}\,|\,\mathbf{o}) = \operatorname*{argmax}_{\mathbf{s}} \frac{p(\mathbf{s},\mathbf{o})}{p(\mathbf{o})} = \operatorname*{argmax}_{\mathbf{s}} p(\mathbf{s},\mathbf{o}) \tag{2.2}$$

The optimal state sequence $\mathbf{s}^*$ may be found efficiently using the Viterbi algorithm (Viterbi, 1967).

2.1.1.1 Limitations of Generative Models

Although hidden Markov models have had many notable successes in language sequencing tasks, they do have restrictions in certain circumstances. One restriction of HMMs relates to the strict conditional independence assumptions they make. For example, as noted in the previous section, in an HMM an observation is dependent only on the state at that sequence position in question. Such strict independence assumptions break our intuitions regarding the dependencies that exist between observation

elements in language. Long-range dependencies, for example, are very difficult to encode in an HMM whilst retaining the ability to train efficiently. A related point here is the ease with which we may specify, in an HMM, arbitrary dependencies between elements in a sentence. For example, it seems intuitive that in a labelling task a state often depends not simply on the identity of the observation elements in a window around that position, but on specific properties they have. To illustrate, in a named entity recognition task we may expect that the current tag labelling decision ought to be influenced not just by the identity of the current word, but by other factors such as the identity of part-of-speech tags in a neighbourhood around the current word, and whether or not the current word is capitalised. Although it is possible to represent these dependencies through an HMM, the state space required often becomes extremely large very quickly, rendering the model intractable.

A second potential restriction of HMMs relates to the fact that they define a joint distribution $p(\mathbf{s}, \mathbf{o})$ over state and observation sequences, thereby implicitly modelling both the conditional distribution $p(\mathbf{s} \mid \mathbf{o})$ and the marginal distribution $p(\mathbf{o})$. In a situation where we have only partially labelled training data, or unlabelled training data, this can be useful because we can still learn the marginal distribution from the data. However, in a situation where we have fully labelled training data and we want to use the HMM to label new unlabelled data, modelling of the entire distribution can make HMMs less discriminative. To address this, a number of approaches have been suggested in the the literature. One possibility is to train the HMM discriminatively, using a discriminative objective function. This typically involves maximising conditional likelihood rather than joint likelihood, resulting in a **conditional maximum likelihood (CML) estimate**, rather than a simple **maximum likelihood (ML) estimate** for the parameters. Klein and Manning (2002) compare ML and CML training regimes using HMMs on a POS tagging task. They find that the CML approach performs marginally better than the ML, having forced the model to focus more on discriminating *between* the labels. In the speech community, CML-trained HMMs are often referred to as **class HMMs (CHMMs)** (Krogh and Riis, 1999). Generally, training HMMs to maximum conditional likelihood requires the use of gradient-descent methods because the objective cannot be solved for the parameters in a closed form (in the case of fully labelled data) and the EM algorithm is not applicable to CML (in the case of partially labelled or unlabelled data). Interestingly, Nadas (1983) shows that if the underlying, "true" distribution lies within the model space, ML can find optimal parameters in the presence of an infinitely large amount of training data. However, in the real-world

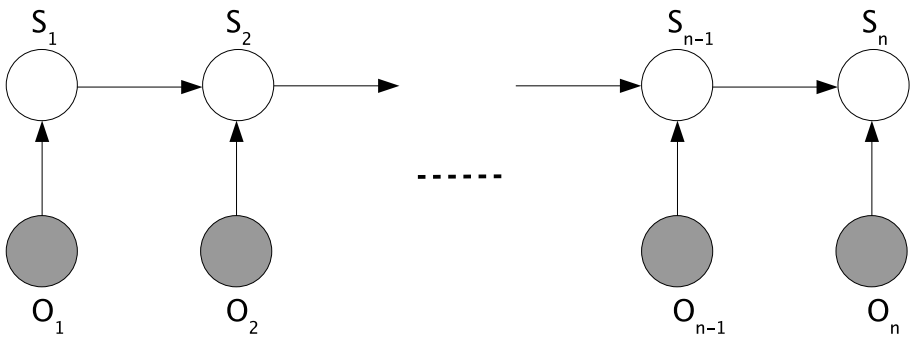

Figure 2.2: CMM graphical model structure.

the "true" distribution is extremely unlikely to lie within the space of HMMs, and the amount of training material is usually relatively limited. Hence discriminative training is often preferred.

Other variants and extensions to the basic HMM structure address our first restriction above, regarding the ease of incorporation of arbitrary local dependencies. Bourlard and Wellekens (1990), for example, show that discriminatively trained HMMs correspond to a particular kind of **multi-layer perceptron (MLP)**. A particular strength of MLPs is their ability to easily incorporate contextual information. Using this capability, Bourlard and Wellekens use MLPs on a speech task, achieving significantly improved classification performance over standard HMMs trained using ML. Other extensions along the CHMM line include **hidden neural networks (HNNs)** (Krogh and Riis, 1999). In HNNs, parameters within a CHMM are replaced by MLPs. Krogh and Riis (1999) use HNNs for the task of recognising broad phoneme classes, and show significant performance gains over standard HMMs on the task.

2.1.2 Discriminative Sequence Models

Discriminative sequence models model the conditional distribution of state sequences given an observation sequence $p(\mathbf{s} \mid \mathbf{o})$ directly, rather than modelling the joint distribution of states and observations and using inference to derive the conditional distribution. In doing so, discriminative sequence models avoid some of the shortcomings of generative sequence models described above. Discriminative sequence models may be either **per-state normalised** or **globally normalised**.

Per-state normalised models assign a probability distribution $p(s_t \mid s_{t-1}, \ldots, \mathbf{o})$ to a state at a particular position in a sequence given some contextual window around

that position. The conditional probability of a state sequence given the observation sequence is then derived (or sometimes approximated) as a function of these local conditional probabilities. Examples of such models include the sequential maximum entropy model and the conditional Markov model (CMM) (Ratnaparkhi, 1996), a special case of which is the maximum entropy Markov model (MEMM) (McCallum et al., 2000). Globally normalised models, by contrast, model the conditional distribution of an entire state sequence given an observation sequence $p(\mathbf{s} \mid \mathbf{o})$ directly, without specifically modelling the marginal state probabilities as part of the process. An example of such a model is a **conditional random field** (Lafferty et al., 2001). As we are considering sequence models used prior to the introduction of CRFs in this section, we will only look at per-state normalised models here.

Figure 2.2 shows the graphical structure of one example of a per-state normalised discriminative model, called a **conditional Markov model (CMM)**. From the diagram we see that the CMM is a discriminative variant of the HMM. Note the reversal of direction of the vertical arrows between the states and the observations. A CMM state is therefore *conditioned* on the observation at the corresponding position in the sequence, in contrast to an HMM where the state is seen as *generating* the observation. Using the chain rule of probability and the conditional independencies in the graph, the distribution for a CMM may be factorised to obtain:

$$p(\mathbf{s}, \mathbf{o}) = p(o_1)\, p(s_1 \mid o_1) \prod_{t=2}^{T} p(o_t)\, p(s_t \mid s_{t-1}, o_t) \tag{2.3}$$

Although this is a joint distribution over states and observations, the graphical structure is conditional in the sense that local conditional probabilities $p(s_t \mid s_{t-1}, o_t)$ appear in the factorisation. In order to obtain the conditional distribution of a state sequence given the observation sequence explicitly, we use:

$$p(\mathbf{s} \mid \mathbf{o}) = \frac{p(\mathbf{s}, \mathbf{o})}{p(\mathbf{o})} \tag{2.4}$$

and note that, because the o_t are marginally independent, we have:

$$p(\mathbf{o}) = \prod_{t=1}^{T} p(o_t) \tag{2.5}$$

and so:

$$p(\mathbf{s} \mid \mathbf{o}) = p(s_1 \mid o_1) \prod_{t=2}^{T} p(s_t \mid s_{t-1}, o_t) \tag{2.6}$$

We can see that, in contrast to the HMM where there is a state-transition conditional probability table $p(s_{t+1} \mid s_t)$ and an observation-emission conditional probability table $p(o_t \mid s_t)$, in a CMM we have a single function $p(s_t \mid s_{t-1}, o_t)$ that describes the probability of transitioning to a state given the previous state and current observation. Note that, as discussed above, the CMM only ever *conditions* on the observation sequence and so does not attempt to model marginal distribution $p(\mathbf{o})$.

A **maximum entropy Markov model (MEMM)** (McCallum et al., 2000) is a specific instance of a CMM where the function $p(s_t \mid s_{t-1}, o_t)$ is split into separate probability distributions, one for each state. As these functions are independent of t, we may instead write $p_s(s' \mid o)$ for the probability of transition to state s' given a state current s and next observation o. In a MEMM each distribution $p_s(s' \mid o)$ is a log-linear model, meaning that it has the form:

$$p_s(s' \mid o) = \frac{1}{Z(s,o)} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(s', o)\right) \tag{2.7}$$

where the λ_k are parameters of the model, the $f_k(s', o)$ are feature functions (described below) and $Z(s,o)$ is a normalising function given by:

$$Z(s,o) = \sum_{s' \in S} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(s', o)\right) \tag{2.8}$$

where S is the set of all states. The use of a log-linear model here allows each distribution to be parameterised by a set of non-independent, overlapping features that describe properties of the observation and state in a given position. This is the model's main advantage over a generative model. Each feature is a function of two arguments, one encoding some property of the current observation and the other specifying the value of the current state. A feature therefore typically has the form:

$$f_{b,s}(s', o) = \begin{cases} 1 & \text{if } b(o) \text{ holds and } s' = s \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

where the function $b(o)$ is a predicate that describes some property of the observation, for example:

$$b(o) = \begin{cases} 1 & \text{if } o \text{ is capitalised} \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

The features do not have to be binary valued, but in most cases they are. The parameters λ_k may be found by maximising the conditional likelihood of state sequences given the observation sequences contained in the training data. McCallum et al. (2000) used **generalised iterative scaling (GIS)** (Darroch and Ratcliff, 1972) for this optimisation, but other numerical methods may be used.

Once trained, an MEMM may be used to label a new observation sequence. Here, as above with the HMM, the task is to identify the most likely state sequence under the model, given the observation sequence. We must find $\mathbf{s}^*$ where:

$$\mathbf{s}^* = \underset{\mathbf{s}}{\operatorname{argmax}}\, p(\mathbf{s} \mid \mathbf{o}) \tag{2.11}$$

This optimal state sequence may be found efficiently using the Viterbi algorithm.

When McCallum et al. (2000) first presented the MEMM as an alternative, improved sequence labelling model over the HMM, they compared its performance to that of two HMM structures on a text segmentation task. Specifically, they used a dataset consisting of documents from seven Usenet multi-part FAQs and attempted to train the classifiers to partition and label subsets of the documents into different categories. The results suggested that the MEMM was indeed capable of outperforming both HMMs on the segmentation task due to its ability to represent a richer set of properties of the observation sequence relevant for labelling decisions. At the time, this result was in line with a view in the community that discriminative classifiers were generally preferable to generative ones. However, Ng and Jordan (2002) showed the situation to be a little more complex. Using naive Bayes and logistic regression as an illustrative generative-discriminative model pair, they demonstrated that the size of the training sample is important in determining the relative performance of each type of model. In general, discriminative models tend to have lower asymptotic error, but the corresponding generative models can in some circumstances approach their higher asymptotic error more quickly.

2.1.2.1 Limitations of Per-state Normalised Discriminative Sequence Models

As we saw in the previous section, an MEMM avoids some of the limitations exhibited by an HMM. One example of this is the MEMM's avoidance of the strong indepen-

dence assumptions made by an HMM. However, MEMMs and other per-state normalised discriminative models do possess certain weaknesses that generative models, such as HMMs, do not. These weaknesses are a product of the conditional independencies defined by their graphical structure, and, in particular, their per-state normalisation property.

Probably the best-known example of this kind of weakness is the **label bias problem**. The term *label bias* was first coined by Lafferty et al. (2001), although the phenomenon was described earlier by Bottou (1991). As a consequence of per-state normalisation, during decoding MEMMs to tend to prefer labellings that involve states (labels) with a small number of successor states, as seen in the training data. In the Viterbi lattice the initial probability mass of 1 is conserved most for these labellings. Any apparently "wrong" decisions early on in the decoding, leading to incorrect partial labellings having a higher probability than the correct labelling at intermediate points in the sequence, are never subsequently "corrected" later in the sequence. This means that there is a labelling bias towards states with one, or a small number of, successors. States with a single successor ignore the observation entirely, while, more generally, states with a small number of successors and a low entropy probability distribution across the successors, tend to be influenced very little by the observation.

Klein and Manning (2002) describe a problem similar in nature to label bias but seen from the point of view of observations rather than labels. They term this phenomenon **observation bias**. As with label bias, observation bias is suffered by per-state normalised discriminative models due to their normalisation properties.

2.2 Conditional Random Fields

Conditional random fields (Lafferty et al., 2001) were introduced to overcome some of the shortcomings of other discriminative sequence models that we looked at above. Being discriminative in nature, CRFs share with MEMMs general advantages over generative models such as avoiding the need to make strict independence assumptions. In addition, CRFs do not suffer the label bias problem because they are not per-state normalised like MEMMs, but are *globally* normalised along the entire sequence (or, more generally, over an arbitrary graphical structure). We explore these properties further in this section.

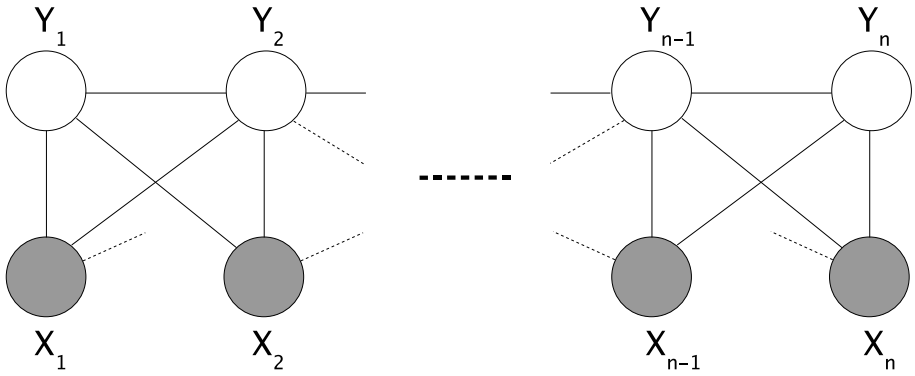

Figure 2.3: Graphical model structure for a linear chain CRF.

2.2.1 Definition and Structure

CRFs are globally normalised conditional probability models. Lafferty et al. (2001) define a CRF as follows:

> Let $G = (V, E)$ be a graph such that $\mathbf{Y} = (\mathbf{Y}_\upsilon)_{\upsilon \in V}$, so that $\mathbf{Y}$ is indexed by the vertices of G. Then $(\mathbf{X}, \mathbf{Y})$ is a *conditional random field* in case, when conditioned on $\mathbf{X}$, the random variables $\mathbf{Y}_\upsilon$ obey the Markov property with respect to the graph, that is
>
> $$p(\mathbf{Y}_\upsilon \,|\, \mathbf{X}, \mathbf{Y}_\omega, \omega \neq \upsilon) = p(\mathbf{Y}_\upsilon \,|\, \mathbf{X}, \mathbf{Y}_\omega, \omega \sim \upsilon),$$
>
> where $\omega \sim \upsilon$ means that ω and υ are neighbours in G.

From this definition we see that a CRF is a Markov random field that is *globally conditioned* on the observation variables $\mathbf{X}$. The definition does not restrict the structure of the graph to a particular form, and arbitrary graphical structures are possible. With more complicated graphs and/or greater connectivity, inference becomes more complex and often some form of scaling is required to keep inference tractable (Cohn, 2006). In this thesis we restrict ourselves to sequence labelling tasks, and so deal with a CRF graphical structure representing a sequence. In this case the graphical structure is usually referred to as a **linear chain CRF**, and is shown in Figure 2.3. As we shall see later, in the case of linear chains efficient dynamic programming algorithms exist for training and decoding.

In Figure 2.3 each observation variable X_i is connected to every other variable in the graph. This is represented by the dotted line leaving each X_i node. As a result of

this structure, no conditional independencies exist between the observation variables. Conditional independencies do exist, however, between the state variables because these have a (first order) linear chain structure. In a general graphical model, the Hammersley-Clifford theorem (Hammersley and Clifford, 1971; Besag, 1974) states that the family of probability distributions respecting the conditional independence semantics of the graph may be parameterised using a set of arbitrary positive potential functions defined on the cliques[3] of the graph. For a clique c with potential function ψ_c, the joint distribution over the variables $\mathbf{Y}$ in the graph is given by:

$$p(y_1,\ldots,y_n) = \frac{1}{Z}\prod_{c\in C}\psi_c(y_c) \tag{2.12}$$

where C is the set of cliques in the graph, y_c are the values of the variables in clique c, and Z is a normalisation constant given by:

$$Z = \sum_{y_1,\ldots,y_n}\prod_{c\in C}\psi_c(y_c) \tag{2.13}$$

For the specific case of the linear chain structure shown in Figure 2.3, the cliques involve edges (Y_{i-1},Y_i) with the Y_i globally conditioned on all the X_i. In addition, Lafferty et al. (2001) suggest potential functions over these cliques that have an exponential form so as to take advantage of properties of the maximum entropy framework. We will adjust the notation to make it consistent with that used earlier for describing HMMs and MEMMs, so the edges become (S_{t-1},S_t), and the observations are denoted by O_t. We will also assume that the chain has T elements and that we add a special placeholder **start** element at the beginning of the chain and similar **stop** element at the end. The element indices therefore run 0 through $T+1$. After this adjustment in notation and using the exponential form for the potential functions, we can see from Equation 2.12 that the distribution for a linear chain CRF is given by:

$$p(\mathbf{s}\mid\mathbf{o}) = \frac{1}{Z(\mathbf{o})}\exp\left(\sum_{t=1}^{T+1}\sum_{k=1}^{K}\lambda_k f_k(s_{t-1},s_t,\mathbf{o},t)\right) \tag{2.14}$$

where $Z(\mathbf{o})$ is a normalising function given by:

$$Z(\mathbf{o}) = \sum_{\mathbf{s}}\left[\exp\left(\sum_{t=1}^{T+1}\sum_{k=1}^{K}\lambda_k f_k(s_{t-1},s_t,\mathbf{o},t)\right)\right] \tag{2.15}$$

[3] A clique is a maximally connected subgraph.

The λ_k are the model **parameters**. They act as co-efficients to the functions f_k, which are **features** defined on subsets of the variables **S** and **O**. They take the same form as those we saw earlier when discussing MEMMs. The difference here is that the argument involving the observations may include all variables **O**. An example of a feature for a POS tagging task, where state variables **S** represent POS tags, could therefore be:

$$f_k(s_{t-1}, s_t, \mathbf{o}, t) = \begin{cases} 1 & \text{if } s_{t-1} = \text{“DET”}, s_t = \text{“NN”} \text{ and } o_t = \text{ “dog”} \\ 0 & \text{otherwise} \end{cases}$$

As before, the features may be real-valued but are often just binary-valued.

We can see from Equation 2.15 that the normalisation for the CRF is *global*, spanning over the entire sequence. It is this property that allows CRFs to solve the label bias problem in a principled way. In per-state normalised conditional models, such as the MEMMs we saw earlier, transitions from a given state effectively compete only against each other and not against transitions from other states (because they sum to 1). In a CRF, however, no such restriction exists. Consequently, in a label bias scenario it is possible with a CRF to have subsequent path down-weighting (like in an HMM). Therefore, with CRFs label bias is avoided.

2.2.2 Training

In this section we consider procedures for training CRFs. The most common method for parameter estimation for CRFs is **conditional maximum likelihood (CML) estimation**. We describe this, and other possible objectives, below.

2.2.2.1 Conditional Maximum Likelihood Estimation

In conditional maximum likelihood estimation we assume that, for a particular parameterisation of the CRF, the parameters λ_k have fixed but unknown values and we seek those parameter values that make the training data most likely. To do this we derive an expression for the conditional likelihood of the training data given a particular model, then attempt to find the model which maximises this function. Let us assume that the training data consists of a set of N fully labelled sequences $D = \left(\mathbf{o}^i, \mathbf{s}^i\right)$ with $i = 1, \ldots, N$. These sequences are a finite sample from the “true” distribution. Denoting the parameters λ_k by a parameter vector $\Theta = (\lambda_1, \ldots, \lambda_K)$, the **conditional likelihood** of the training data is given by:

$$\mathscr{L}(\Theta) = \prod_{\mathbf{o},\mathbf{s}} p(\mathbf{s} \mid \mathbf{o}, \Theta)^{c(\mathbf{o},\mathbf{s})} \tag{2.16}$$

where $p(\mathbf{s} \mid \mathbf{o}, \Theta)$ is the model distribution and $c(\mathbf{o}, \mathbf{s})$ is a count of the number of times the configuration $(\mathbf{o}, \mathbf{s})$ occurs in the training data. We define the **empirical distribution** (or training data distribution) $\tilde{p}(\mathbf{o}, \mathbf{s})$ as:

$$\tilde{p}(\mathbf{o}, \mathbf{s}) = \frac{c(\mathbf{o}, \mathbf{s})}{N} \tag{2.17}$$

Each term in the product in the conditional likelihood function in Equation 2.16 is non-negative, so we could equivalently maximise the alternative function:

$$L(\Theta) = \prod_{\mathbf{o},\mathbf{s}} p(\mathbf{s} \mid \mathbf{o}, \Theta)^{\tilde{p}(\mathbf{o},\mathbf{s})} \tag{2.18}$$

This function makes subsequent derivation easier. Therefore, we look for the parameter vector that maximises this function, the optimum value being the conditional maximum likelihood estimate Θ_{CML}:

$$\Theta_{\text{CML}} = \operatorname*{argmax}_{\Theta} L(\Theta) \tag{2.19}$$

To further simplify the derivation, it is sensible to maximise the logarithm of $L(\Theta)$ rather than $L(\Theta)$ itself. We can do this because the logarithm is a monotonically increasing function for positive real numbers. Hence the task is to find the parameter vector Θ that maximises the conditional log-likelihood, given by:

$$\Lambda(\Theta) = \log L(\Theta) = \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o}, \mathbf{s}) \log p(\mathbf{s} \mid \mathbf{o}, \Theta) \tag{2.20}$$

In the case of a CRF, as we saw above, the distribution is given by:

$$p(\mathbf{s} \mid \mathbf{o}, \Theta) = \frac{1}{Z(\mathbf{o})} \exp\left(\sum_{t=1}^{T+1} \sum_{k=1}^{K} \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t) \right) \tag{2.21}$$

Contracting $\sum_k \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t)$ to $\lambda.\mathbf{f}$, the expression for the conditional log-likelihood becomes:

$$\Lambda(\Theta) = \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \left[\sum_{t=1}^{T+1} \lambda.\mathbf{f} \right] - \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \log Z(\mathbf{o}) \tag{2.22}$$

Because the summand in the second term is a function of the observation sequence only, the conditional log-likelihood can be further simplified to:

$$\Lambda(\Theta) = \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \left[\sum_{t=1}^{T+1} \lambda.\mathbf{f} \right] - \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \log Z(\mathbf{o}) \tag{2.23}$$

This is the **objective function** to be maximised under conditional maximum likelihood parameter estimation. An important property of this function is that it is **convex** so has a unique maximum.

At this point it is tempting to take the derivative of the objective with respect to each parameter, set each derivative to zero and try to solve for the parameter values. Unfortunately, the resulting equations are implicit in the parameters and in general a closed form solution cannot be found. In fact, because the CRF has a log-linear form, when we carry out the above procedure we obtain the maximum entropy feature constraints – that the expected values of the features under the model and empirical distributions are equal. This is related to the fact that the maximum likelihood model across the space of exponential models on a given training set is also the unique model that maximises the entropy across the space of models that obey the feature constraints on the training set. To see this, we differentiate with respect to parameter λ_k to obtain:

$$\frac{\partial \Lambda(\Theta)}{\partial \lambda_k} = \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \sum_{t=1}^{T+1} f_k(s_{t-1}, s_t, \mathbf{o}, t) - \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \frac{1}{Z(\mathbf{o})} \frac{\partial Z(\mathbf{o})}{\partial \lambda_k} \tag{2.24}$$

which simplifies to:

$$\begin{aligned} \frac{\partial \Lambda(\Theta)}{\partial \lambda_k} &= \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \sum_{t=1}^{T+1} f_k(s_{t-1}, s_t, \mathbf{o}, t) - \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o})\, p(\mathbf{s} \mid \mathbf{o}, \Theta) \sum_{t=1}^{T+1} f_k(s_{t-1}, s_t, \mathbf{o}, t) \\ &= E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o},\Theta)}[f_k] \end{aligned} \tag{2.25}$$

Because in general we cannot solve for the parameters of the conditional maximum likelihood model analytically, we must turn to numerical algorithms to find them. We describe these algorithms later, in section 2.2.2.3. Typically a numerical optimisation algorithm requires the calling process to supply the value of the objective function

and its gradient at each iteration (and in some cases a representation of the second derivatives). This means that the expression on the right-hand side of Equation 2.25 must be evaluated on each iteration, for the parameter vector representing the current point in the parameter space being searched. The first expression is relatively easy to evaluate and is not iteration-dependent, so can be done once, offline. The second expression is more troublesome, and warrants a little investigation. The task, then, is to evaluate:

$$E_{p(\mathbf{s}|\mathbf{o},\Theta)}[f_k] = \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o})\, p(\mathbf{s} \mid \mathbf{o}, \Theta) \sum_{t=1}^{T+1} f_k(s_{t-1}, s_t, \mathbf{o}, t) \tag{2.26}$$

At first sight it may appear that the evaluation of this expression is intractable for tasks that involve a large number of states. This is because it requires the summand:

$$\tilde{p}(\mathbf{o})\, p(\mathbf{s} \mid \mathbf{o}, \Theta) \sum_{t=1}^{T+1} f_k(s_{t-1}, s_t, \mathbf{o}, t) \tag{2.27}$$

to be calculated for every possible state sequence for each observation sequence present in the training data. However, an efficient dynamic programming algorithm exists which allows us to circumvent this potential intractability. In order to understand how this works, let us first rewrite the expression to be evaluated (dropping the explicit reference to the dependence on the parameters Θ):

$$\begin{aligned}
\sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o})\, p(\mathbf{s} \mid \mathbf{o}) \sum_{t=1}^{T+1} f_k(s_{t-1}, s_t, \mathbf{o}, t) &= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{t=1}^{T+1} \sum_{\mathbf{s}} f_k(s_{t-1}, s_t, \mathbf{o}, t)\, p(\mathbf{s} \mid \mathbf{o}) \\
&= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{t=1}^{T+1} \sum_{s',s} f_k(s_{t-1} = s', s_t = s, \mathbf{o}, t) \\
&\quad \times p(s_{t-1} = s', s_t = s \mid \mathbf{o}) \\
&= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{t=1}^{T+1} \sum_{s',s} f_k(s_{t-1} = s', s_t = s, \mathbf{o}, t) \\
&\quad \times \frac{\alpha_{t-1}(s' \mid \mathbf{o})\, M_t(s', s \mid \mathbf{o})\, \beta_t(s \mid \mathbf{o})}{Z(\mathbf{o})}
\end{aligned} \tag{2.28}$$

where the α_t are **forward vectors**, the β_t are **backward vectors** (both defined below) and the M_t are matrices defined on each clique in the chain. Specifically, on the tth clique in the chain the matrix M_t is defined by:

$$M_t\left(s', s \mid \mathbf{o}\right) = \exp\left(\sum_k \lambda_k f_k\left(s_{t-1} = s', s_t = s, \mathbf{o}, t\right)\right) \tag{2.29}$$

Each matrix is square, with dimension equal to the number of states. The (s', s) element contains the value of the potential function on the tth clique when $s_{t-1} = s'$ and $s_t = s$. Hence the (s', s) element of the product of these matrices between positions t' and t in the sequence represents the sum of the product of the potentials along all paths between positions t' and t that have $s_{t'} = s'$ and $s_t = s$. A special case of this is the product of all matrices in the sequence from cliques 1 to $T+1$, $\prod_{t=1}^{T+1} M_t(\mathbf{o})$. This represents the sum of the product of the potentials along all paths, for all possible values of s_0 and s_{T+1}, the placeholder elements. In particular, the normalising function along the sequence is given by that matrix element that corresponds to $s_0 = \textbf{start}$ and $s_{T+1} = \textbf{stop}$:

$$Z(\mathbf{o}) = \left[\prod_{t=1}^{T+1} M_t(\mathbf{o})\right]_{\textbf{start, stop}} \tag{2.30}$$

The α_t vectors are defined recursively via:

$$\alpha_t(\mathbf{o})^T = \alpha_{t-1}(\mathbf{o})^T M_t(\mathbf{o}) \tag{2.31}$$

The base case is given by:

$$\alpha_0(s \mid \mathbf{o}) = \begin{cases} 1 & \text{if } s = \textbf{start} \\ 0 & \text{otherwise} \end{cases} \tag{2.32}$$

Hence $\alpha_t(s \mid \mathbf{o})$ contains the product of the potentials along all paths starting in the **start** state at s_0 and ending in the state s at s_t. The base case $\alpha_0(\mathbf{o})$ therefore acts like an initial probability vector in an HMM. Note the distinction here between an alpha vector and the value of a particular component of the vector: the alpha vector on clique t is denoted by $\alpha_t(\mathbf{o})$, but the sth component of the vector is denoted by $\alpha_t(s \mid \mathbf{o})$. The same holds for beta vectors $\beta_t(\mathbf{o})$ and the matrices $M_t(\mathbf{o})$.

The β_t vectors are defined in a recursive manner, similar to the alpha vectors:

$$\beta_t(\mathbf{o}) = M_{t+1}(\mathbf{o})\,\beta_{t+1}(\mathbf{o}) \tag{2.33}$$

The base case is given by:

$$\beta_{T+1}(s \mid \mathbf{o}) = \begin{cases} 1 & \text{if } s = \textbf{stop} \\ 0 & \text{otherwise} \end{cases} \tag{2.34}$$

Hence $\beta_t(s \mid \mathbf{o})$ contains the product of the potentials along all paths ending in the **stop** state at s_{T+1} and starting in the state s at s_t.

Because the joint probability $p(s_{t-1} = s', s_t = s \mid \mathbf{o})$ may be expressed using the matrices $M_t(\mathbf{o})$, we may use dynamic programming to efficiently compute the feature expectations under the model. This requires one forward pass to calculate the $\alpha_t(\mathbf{o})$ vectors, and one backward pass to calculate the $\beta_t(\mathbf{o})$. In calculating the $\beta_t(\mathbf{o})$ vectors the normalising function may be obtained by forming one additional product:

$$\begin{aligned} Z(\mathbf{o}) &= \left[\prod_{t=1}^{T+1} M_t(\mathbf{o})\right]_{\textbf{start, stop}} \\ &= [M_1(\mathbf{o})\,\beta_1(\mathbf{o})]_{\textbf{start}} \\ &= [\beta_0(\mathbf{o})]_{\textbf{start}} \end{aligned} \tag{2.35}$$

This may alternatively be obtained by calculating one additional α vector, $\alpha_{T+1}(\mathbf{o})$:

$$Z(\mathbf{o}) = [\alpha_{T+1}(\mathbf{o})]_{\textbf{stop}} \tag{2.36}$$

It is possible to modify the CML estimation method above to include regularisation in the form of a prior distribution over the model parameters. We will discuss this in more detail later in the chapter, in section 2.3.1, when we describe overfitting reduction schemes.

When describing the training procedure earlier we assumed that the parameter vector is updated on each iteration using the expected values in Equation 2.25. These expected values are calculated using the entire training dataset. Therefore we require one complete pass through the dataset for each iteration. This makes the procedure described above a **batch** algorithm. It is possible, however, to employ an **online** algorithm instead. With this, the gradient would be calculated, and the parameter vector updated, after passing through only a subset of the training data. The extreme case would be an parameter vector update for every training instance, i.e. every sequence in the training data. For some datasets a single training instance may be almost as informative as the whole dataset in terms of gradient calculation, so employing an online

algorithm can sometimes lead to much faster convergence. In other cases the opposite is true, where the added expense of calculating the parameter vector for each training instance makes the online algorithm converge more slowly.

In addition to the frequency of parameter vector updates, other modifications to the training algorithm are possible. One example relates to the calculation of the expected feature count under the model, $E_{p(\mathbf{s}|\mathbf{o},\Theta)}[f_k]$, in Equation 2.25. This is calculated over the entire distribution, that is, over all labellings for each sequence. However, we could instead replace this with the feature count for only the *most likely* labelling under the model. In doing so we assume that most of the probability mass lies in this labelling, and the rest of the distribution can be ignored. This is called making a **Viterbi assumption**. The usefulness of this approach involves a trade-off between the exactness of the gradient calculation and the training time. For the experiments we present later in the thesis, training time is reasonable even with the full expected value calculation. Hence we use Equation 2.25 for the gradient vector calculation.

2.2.2.2 Other Objective Functions

In the previous section we showed how to estimate the parameters of a CRF using conditional maximum likelihood estimation. However, other choices of objective function exist. The most notable study to date investigating alternative objective functions for CRFs is that of Altun et al. (2003). We briefly describe their work in this section.

Altun et al. argue that sequence labelling tasks in NLP vary widely in nature with respect to the style of the labelling problem. Some tasks, such as in named entity recognition, involve labels that range over many elements in a sequence. Other tasks, such as part-of-speech tagging, involve labels that apply only to single elements. In addition, the statistical noise associated with a task varies with both the task itself and the specific corpus used for training and testing. As a result, Altun et al. conjecture that using different objective functions that are tailored to the task in hand may result in better classifiers. They propose two dimensions along which objective functions may differ, one representing the *numerical scale* on which the objective is measured (exponential versus logarithmic), the other the *range* of elements within a sequence over which the objective is defined (pointwise versus sequential). Taking the cross-product of the values along these two dimensions, Altun et al. define four objective functions, one of which is the standard conditional log-likelihood function we discussed earlier. The objectives are compared on two tasks: part-of-speech tagging (on sections of the Penn Treebank (Marcus et al., 1993)) and named entity recognition (on Spanish news-

wire articles provided from the CoNLL 2002 shared task (Kim Sang, 2002)). Altun et al. found that the choice of objective function made very little difference to performance on either task, and concluded that other factors, such as the choice of feature set, are more important. For the experiments we present in this thesis, we only use an objective function based on the standard CML estimation method.

2.2.2.3 Optimisation of the Objective Function

Until recently the most popular techniques for parameter estimation with maximum entropy models were iterative scaling algorithms: **generalised iterative scaling (GIS)** (Darroch and Ratcliff, 1972) and **improved iterative scaling (IIS)** (Della Pietra et al., 1997). However, recent work by Malouf (2002) showed that iterative scaling algorithms underperform a number of first and second order numerical optimisation algorithms for parameter estimation for conditional maximum entropy models on various NLP tasks. Prompted by this result, Wallach (2002) undertook a similar study for parameter estimation for CRFs. Her results broadly supported those of Malouf, and subsequent studies have done the same (Sha and Pereira, 2003).

In general, the most popular algorithms in use today for parameter estimation for CRFs are related to the **limited memory variable metric (LMVM)** (or **L-BFGS**) method (Byrd et al., 1995). This method uses a sparse representation of the Hessian matrix (the matrix of second derivatives of the objective function) that requires storage of only a limited history of the incremental values of the objective function and its first derivatives. This history is typically less than 20 steps. Other methods that may be used are conjugate gradient methods, which only represent the first derivatives and not the Hessian. Examples include **Fletcher-Reeves** (Fletcher and Reeves, 1964) and **Polak-Ribi*è*re-Positive** (Polak and Ribière, 1969). These methods have generally been found to underperform the LMVM method, but often outperform iterative scaling. Sha and Pereira (2003) use a pre-conditioner with their conjugate gradient method, whereby the method is accelerated by linearly transforming the variables. Other second order methods that might theoretically be candidates include Newton's method and quasi-Newton methods. However, these either require the inverse Hessian to be explicitly calculated, or at least a dense approximation of it to be calculated from first derivatives. Consequently, for typical NLP problems, which often include millions of parameters, these approaches are too demanding from a computational and/or storage point of view. Currently, therefore, LMVM is the method of choice for CRF parameter estimation. For all our experiments we use the LMVM method for optimisation of the objective.

2.2.3 Decoding

Once a CRF has been trained, it may be used to decode (or label) other, unseen observation sequences. Referring back to the discussion in section 2.2.2.1 when we considered CRF parameter estimation, the probability of a particular state sequence given an observation sequence is simply the product of the potentials on each clique for the states in question, divided by the normalising function. The product potential may be found by taking the product of the relevant entries from the $M_t(\mathbf{o})$ matrices along the sequence. Hence:

$$p(\mathbf{s} \mid \mathbf{o}) = \frac{\prod_{t=1}^{T+1} M_t(s_{t-1}, s_t \mid \mathbf{o})}{Z(\mathbf{o})} \tag{2.37}$$

The normalising function is not dependent on the state sequence, so the optimal sequence (the one that has highest probability under the model, given the observation sequence) can be found using a standard Viterbi algorithm (Viterbi, 1967).

To find the probability of a specified labelling, or to find the Viterbi labelling, only one sweep through the lattice is required, either left-to-right or right-to-left. However, if we conduct a sweep in both directions we have enough information to calculate the pointwise marginal label distributions $p(s_t \mid \mathbf{o})$. To see this, glance back to Equation 2.28. From the expansion on the right-hand side, we can see that:

$$p\left(s_{t-1} = s', s_t = s \mid \mathbf{o}\right) = \frac{\alpha_{t-1}(s' \mid \mathbf{o}) M_t(s', s \mid \mathbf{o}) \beta_t(s \mid \mathbf{o})}{Z(\mathbf{o})} \tag{2.38}$$

Now, the pointwise marginal distributions are given by summing over the previous label in the corresponding pairwise marginal distribution. So:

$$\begin{aligned} p(s_t = s \mid \mathbf{o}) &= \sum_{s'} p\left(s_{t-1} = s', s_t = s \mid \mathbf{o}\right) \\ &= \sum_{s'} \frac{\alpha_{t-1}(s' \mid \mathbf{o}) M_t(s', s \mid \mathbf{o}) \beta_t(s \mid \mathbf{o})}{Z(\mathbf{o})} \end{aligned} \tag{2.39}$$

But using Equation 2.31 we can contract the first two terms in the numerator on the right-hand side to give:

$$p(s_t = s \mid \mathbf{o}) = \frac{\alpha_t(s \mid \mathbf{o}) \beta_t(s \mid \mathbf{o})}{Z(\mathbf{o})} \tag{2.40}$$

Therefore, the pointwise marginal label distributions are given by the product of the relevant entries in the alpha and beta vectors on the clique in question, divided by the normalising function. We will be using pointwise marginal distributions in Chapter 6.

2.2.4 Applications

In their original CRF paper, Lafferty et al. (2001) introduced a linear chain CRF and demonstrated its effectiveness on a synthetic dataset. Subsequently, linear chain CRFs were applied effectively to a wide range of real sequence labelling tasks in NLP. These include named entity recognition (McCallum and Li, 2003; Smith and Osborne, 2005, 2006), noun phrase chunking and shallow parsing (Sha and Pereira, 2003), information extraction from research papers (Peng and McCallum, 2004; Pinto et al., 2003), and language modelling (Roark et al., 2004). CRFs have also been developed for more complex graphical structures, including trees and loopy graphs. Inference on these more general structures involves a generalised form of the forward-backward procedure we saw in section 2.2.2.1, usually employing some form of **belief propagation**. Such CRFs have been applied to problems with more complex dependencies than can easily be represented by a linear chain. Examples from NLP include semantic role labelling (Cohn and Blunsom, 2005; Tang et al., 2006) and co-reference resolution (Wellner et al., 2004; McCallum and Wellner, 2004).

In addition to the variety of tasks, CRFs have also been employed for sequence labelling tasks in a number of other languages. These include NER in German, Japanese and Hindi (McCallum and Li, 2003; Asahara and Matsumoto, 2003; Li and McCallum, 2003), word segmentation in Chinese (Peng et al., 2004), text chunking in Korean (Lee et al., 2005), and morphological analysis in Thai (Kruengkrai et al., 2006).

Outside NLP, CRFs have been used for many different structured labelling tasks in a number of different domains. Examples include *computer vision*, where they have been used for object detection (Torralba et al., 2005), object recognition (Quattoni et al., 2005), object segmentation (Wang and Ji, 2005), and modelling human motion and gestures (Sminchisescu et al., 2005; Wang et al., 2006); *speech*, where they have been used to predict pitch accents (Gregory and Altun, 2004); and *genetics*, where they have been used to locate introns and exons in DNA sequences (Culotta et al., 2005).

2.3 Overfitting

The term *overfitting* describes the situation where a model, during training, overly fits to the idiosyncratic or noisy properties of the training sample, rather than just the systematic patterns implicit in the underlying data. In general this can occur when a model has a relatively large number of degrees of freedom in comparison to the size and complexity of the training sample. It can therefore take place either when the model has too many parameters, the sample is too small, or both. With CRFs often the overfitting problem takes place in the second of these scenarios, where the model has roughly the number of degrees of freedom thought necessary to represent the most important aspects of the underlying distribution, but where the dataset is too small for all facets of the distribution to be observed in sufficient detail to be modelled accurately. The result is that some degrees of freedom fit to aspects of the training sample that represent noise, and do not reflect the underlying data from which the sample is drawn.

A very clear, simple example illustrating overfitting is given by Bishop (1995). He describes a regression scenario where some data points have been sampled from a function containing a sine wave and a noise term. The presence of the noise term means that the data points sampled do not lie exactly on the curve, but nevertheless trace out the general shape of the curve. The aim is to model the underlying function as well as possible given only the relatively small number of sample data points. If a polynomial is used as a model, the number of degrees of freedom of the model is represented by the order of the polynomial. The greater the order, the larger the number of free co-efficients to fit to the sample data. For polynomials of low order, there are not enough degrees of freedom to model the underlying function from which the data is sampled. For example, for a polynomial of first order the resulting straight line is a very bad approximation to the underlying function except, possibly, in the region of some of the data points. Conversely, for a polynomial of very high degree (much larger than the number of data points in the sample, for example), the resulting curve fits all the data points exactly, but, in order to do this, it has very high curvature in between the data points, and consequently generalises very badly to points other than those in the sample. The “correct” choice is a polynomial of intermediate order, which has enough degrees of freedom to capture the general shape of the sine wave as represented through the noisy sample, but not so many degrees of freedom that the noise itself is modelled.

This general concept of trying to find the "best" model can also be illustrated from the viewpoint of a **bias-variance decomposition** (Geman et al., 1992). In the context of the Bishop regression example above, the bias-variance decomposition quantifies the effectiveness of the learned function (the polynomial) as a predictor of the function being modelled (the sine wave). This "effectiveness" is usually represented as an expectation across all values of the output variable for a given value of the input variable, across all values of the input variable, and across all training samples that could have been used to arrive at the learned function (as different training samples typically result in different learned functions). The expectation decomposes into three terms:

1. **noise term**: this term quantifies the inherent uncertainty, or noise, in the data. In the Bishop example this would be the variance of the output variable (the noisy sine function) given the input variable, averaged across all values of the input variable.

2. **bias term**: this term quantifies how closely the learned function can model the underlying function. In the Bishop example this term would be the error between the value of the average learned function (taken across all training samples) at a given value of the input variable, and the average value of the output variable at that value of the input variable, averaged across all values of the input variable.

3. **variance term**: this term quantifies the amount by which the learned function is sensitive to the variability in the training sample used to train it. In the Bishop example this term would be the variance in the value of the learned function (taken across all training samples) for a given value of the input variable, averaged across all values of the input variable.

Models which have a relatively large number of degrees of freedom (like polynomials of high order in the Bishop example) will in general model the underlying function better, so will tend to have a lower bias term. However, the extra degrees of freedom these models contain will often cause them to overfit the details of the training data, making the models sensitive to the training sample. Such models will therefore generally have a higher variance term. Conversely, simpler models with fewer degrees of freedom (lower order polynomials in the Bishop example) will tend to have a larger bias term because they are not, in general, able to model the underlying function as closely. They do, however, often have a lower variance term because they are not as sensitive to the details of the training sample used to train them. Finding the "best"

model therefore involves a trade-off between the bias and variance terms in the bias-variance decomposition. Earlier, when we were suggesting the "best" model would be a polynomial of intermediate order, we were implicitly considering this bias-variance trade-off.

The example from Bishop described above involves a regression problem, but a similar concept applies to the models we described earlier in the thesis: HMMs, MEMMs and CRFs. CRFs particularly have the tendency to overfit severely on some datasets. The reason why they tend to overfit to a greater degree than similar models, such as MEMMs, relates to the normalisation. Suppose we have an MEMM and a CRF with the same feature set. Both models have the same number of parameters, but, as we saw in section 2.1.2, the MEMM is pointwise normalised whereas a CRF is globally normalised along a sequence. Consequently the MEMM must satisfy a normalisation constraint for each state being transitioned from. Intuitively, the parameters trade-off against each other only at a point. With a CRF, however, the normalisation is global so parameters can trade-off against each other along the entire sequence. Consequently the CRF is more flexible, having greater power to fit to the intricacies of the dataset, but therefore also has a greater propensity to *overfit*.

2.3.1 Overfitting Reduction

Various methods may be used to address the overfitting problem described in the previous section. The approach that is often taken is to deal directly with the symptoms of overfitting. An example of this is given by Bishop (1995) for the function approximation problem described above. A symptom of the overfitting in that example is the very high curvature of the function in between the sample data points. This is an undesirable property for an approximating function, which we would hope would be continuous and smooth between the data points in the sample. One way to address this, therefore, is to add a penalty term to the objective function that explicitly penalises large curvature. Bishop suggests a term that is proportional to the integral of the square of the second derivative of the function over the relevant interval. This encourages the function to have low curvature but still fit reasonably well to the sample data points.

In the case of CRFs, and log-linear models in general, a typical symptom of overfitting is parameter values with very large magnitude. We may apply an engineering approach to the overfitting problem in these models that is analogous to the large curvature penalty term in Bishop's function approximation example. One way to do this is

to add a penalty term to the objective function, typically the conditional log-likelihood, that directly discourages parameter values of large magnitude. An example of a such a penalty would be a term that sums together the squares of the parameter values, i.e. $\sum_{k=1}^{K} \lambda_k^2$. Assuming that we are maximising an objective function, we would then *subtract* this term from the existing objective. More generally, we could include a term of the form $-\sum_{k=1}^{K} (\lambda_k - c_k)^2$, which discourages each parameter value λ_k from straying very far from the value c_k. The c_k values allow us to encode our prior beliefs about where "good" parameter values should lie.

The approach to overfitting described in the previous two paragraphs we loosely termed an "engineering" approach because we directly addressed the symptoms of overfitting via the inclusion of a penalty term. The approach did not have an explicit statistical motivation. However, the same idea may also be viewed from a Bayesian probabilistic standpoint. Taking a Bayesian approach, we think of the model parameters as random variables and endow them with a **prior distribution**. This distribution encodes our *prior beliefs* about which models are more likely than others, and is typically parameterised by one or more **hyperparameters**. Under the Bayesian paradigm, we update the prior distribution on the basis of the information contained in the training data, to form the **posterior distribution**. The posterior distribution therefore represents our updated beliefs about the which models are more likely than others, having observed the training data. Under a proper Bayesian treatment, we would then use the posterior distribution to calculate various statistics of interest to us, typically calculating expected values over that distribution. In principle we can continue the prior concept further by thinking of the hyperparameters in the prior distribution as random variables in their own right, and endowing them with distributions of their own. These distributions are often referred to as **hyperpriors**, and they are typically parameterised by further hyperparameters. It is clear that such a **Bayesian hierarchical model** quickly leads to a very large number of hyperparameters, and involves the evaluation of complicated integrals in calculating statistics of interest. A simpler approach is to terminate the hierarchy at the first level, thus dealing only with a single set of hyperparameters. A further simplification is to use a single representative model from the posterior distribution, rather than taking averages over the whole distribution. This leads to the idea of a **maximum a posteriori (MAP)** model.

With the MAP model, or **MAP estimate**, we take a single point estimate of the model parameters that corresponds to the **mode** of the posterior distribution. Chen and Rosenfeld (1999) propose this method as a way of smoothing maximum entropy

models, and use a Gaussian distribution as the choice of prior. The same idea can be applied to CRFs. To see this, we start with a prior distribution $p(\Theta)$ and seek an expression for the posterior distribution $p(\Theta \mid D)$, having observed training data D. The posterior may be expressed in terms of the conditional likelihood function and prior distribution using Bayes' rule:

$$p(\Theta \mid D) = \frac{p(D \mid \Theta)\, p(\Theta)}{\int p(D \mid \Theta)\, p(\Theta)\; d\Theta} \tag{2.41}$$

The integral in the denominator is over the space of all models, as defined by the parameter vector Θ. The MAP estimate Θ_{MAP}, being the mode of the posterior distribution, is therefore given by:

$$\begin{aligned} \Theta_{\text{MAP}} &= \operatorname*{argmax}_{\Theta}\; p(\Theta \mid D) \\ &= \operatorname*{argmax}_{\Theta}\; p(D \mid \Theta)\, p(\Theta) \\ &= \operatorname*{argmax}_{\Theta}\; \log\left[p(D \mid \Theta)\, p(\Theta)\right] \\ &= \operatorname*{argmax}_{\Theta}\; \left[\log p(D \mid \Theta) + \log p(\Theta)\right] \end{aligned} \tag{2.42}$$

Hence the new objective function consists of the conditional log-likelihood, as before, along with the additional term $\log p(\Theta)$.

Now, it is common to assume that the parameters, as random variables, are independent of each other. Under this assumption, the prior joint density may be factorised into the product of the marginal densities, as follows:

$$p(\Theta) = \prod_{k=1}^{K} p_k(\lambda_k) \tag{2.43}$$

Taking the logarithm of both sides gives:

$$\log p(\Theta) = \sum_{k=1}^{K} \log p_k(\lambda_k) \tag{2.44}$$

In addition, we assume the parameters are identically distributed and each parameter is endowed with a Gaussian distribution. We therefore have:

$$p(\lambda_k) = \frac{1}{\sigma_k \sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma_k^2}\left(\lambda_k - \mu_k\right)^2\right) \tag{2.45}$$

for hyperparameters μ_k and σ_k^2, the means and variances respectively. Taking the logarithm of both sides and substituting back into Equation 2.44 gives:

$$\begin{aligned}\log p(\Theta) &= \sum_{k=1}^{K}\left[-\frac{1}{2}\log 2\pi - \log\sigma_k - \frac{1}{2\sigma_k^2}(\lambda_k-\mu_k)^2\right] \\ &= -\frac{K}{2}\log 2\pi - \sum_{k=1}^{K}\log\sigma_k - \frac{1}{2}\sum_{k=1}^{K}\left(\frac{\lambda_k-\mu_k}{\sigma_k}\right)^2 \end{aligned} \tag{2.46}$$

Note that the first two terms in this expression are independent of the λ_k, so can be ignored. We therefore arrive at an additional term in the objective:

$$-\frac{1}{2}\sum_{k=1}^{K}\left(\frac{\lambda_k-\mu_k}{\sigma_k}\right)^2 \tag{2.47}$$

This term is very similar in form to the penalty we suggested earlier when we took an engineering approach to the overfitting problem.

Having defined the hyperparameters μ_k and σ_k, we need a way to set their values. Different possibilities exist here. One method is to derive the values of the hyperparameters from the training data. This general approach is often referred to as **Empirical Bayes** (Carlin and Louis, 2000; Gelman et al., 2003), and includes a range of methods. One way to employ Empirical Bayes is to approximate the marginal distribution (the denominator in Equation 2.41) as a function of the hyperparameters only. The training data is then used to generate maximum likelihood estimates of parameters of the marginal distribution (such the mean and variance), thus giving estimates for the hyperparameters themselves. However, in typical NLP problems when applying the MAP approach to CRFs (and other log-linear models), it is more common to set the values of the hyperparameters using a held-out dataset, after certain simplifications have been made. Allocating an independently variable hyperparameter σ_k for each parameter λ_k, and searching the for the optimal set of σ_k values using held-out data is usually computationally too costly. Therefore, it is common to constrain all σ_k to be equal to a single adjustable variable σ. In addition, it is typical set the mean of the distribution (the μ_k) to the zero vector. Having made these simplifications, the single independent variable σ can easily be found using a held-out dataset. This may be achieved either by manual tuning using a range of values, or by employing an automated search process such as a line search. Either way, using the simplifications above, a typical objective function for a CRF with MAP estimation becomes:

$$L(\Theta) - \frac{1}{2\sigma^2}\sum_{k=1}^{K}\lambda_k^2 \tag{2.48}$$

The additional term makes a contribution to the gradient of the objective function. The new derivative, with respect to parameter λ_k, is given by:

$$E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o},\Theta)}[f_k] - \frac{\lambda_k}{\sigma^2} \tag{2.49}$$

Clearly, the additional term in the gradient is very inexpensive to evaluate, and does not require an elaborate dynamic programming approach in contrast to the second term from the original expression. The presence of the additional prior term does not change the convexity of the objective function, so the maximum still occurs at a unique point in the parameter space.

In our description of MAP estimation here we have used a Gaussian distribution as the prior, but it other choices are possible. Peng and McCallum (2004) compare a Gaussian to Laplacian and Hyperbolic priors on a task involving extraction of information from research papers. They conclude that the Gaussian is in general a better choice than the other two priors. We conduct similar experiments on two sequence labelling tasks in Chapter 5, but reach alternative conclusions.

With the MAP estimation method above we collapse the posterior distribution to a point estimate, the MAP estimate. As we described above, a full Bayesian treatment would instead use the posterior distribution during decoding to average across a spectrum of models. Qi et al. (2005) attempt to do something along these lines with their **Bayesian conditional random field (BCRF)** model. They use the Power Expectation Propagation (Power EP) (Minka, 2004) method, an extension of Minka's EP (Minka, 2001), to calculate the posterior distribution during training and decoding. They show that BCRFs can outperform standard CRFs trained with CML and MAP on synthetic data and a real FAQ labelling task. Although this model represents a step forward in principle, there are some drawbacks. For example, the BCRF does still involve some approximations, so does not represent a genuine Bayesian treatment of the problem. In addition, BCRFs can be computationally more complex than standard CRFs trained with CML or MAP, depending on the training regimes used for each. Lastly, it is not clear how well the BCRF approach scales with the complexity of the problem or size of the dataset.

In the domain of neural networks, ensembles have been shown in many circumstances to reduce generalisation error over that of the models making up the ensemble. As result, we can think of ensembles as another approach to reducing overfitting, and, indeed this is the idea that underlies our work with **logarithmic opinion pools** of CRFs in later chapters. We will discuss previous use of ensembles a little more in Chapter 6, when we introduce LOPs.

2.4 Summary

In this chapter we have provided a general background to the material presented in the rest of the thesis, the intention being to cover the main concepts required to understand later chapters. Specifically, we have done the following:

- Situated CRFs with respect to previous, comparable sequence labelling models. In particular, we looked at the strengths and weaknesses of generative sequence models, such as HMMs, and saw how these models were superseded by discriminative sequence models such as MEMMs. We also briefly looked at the shortcomings of discriminative models, such as the *label bias problem*.

- Described CRFs in detail, covering the definition of a CRF and its structure as a graphical model, training and decoding algorithms for CRFs, and the applications of CRFs to date, both in NLP and in other domains.

- Considered the phenomenon of overfitting, and methods that can be employed to reduce the effect.

In the next chapter we move on to describe the experimental setup used for the experiments conducted in the thesis.

Chapter 3

Experimental Setup

In this chapter we describe the conditions used to conduct the experiments presented in this thesis. By doing this we aim to make the our results easy to reproduce. The setup is essentially the same for most of the experiments. The experiments themselves, and the results, are covered in more detail later in chapters 5, 6, 7 and 8. This chapter, more specifically, covers the following:

- The two labelling tasks we use for our experiments: **named-entity recognition** and **part-of-speech tagging**.
- The experimental pipeline and software architecture.
- The measures we use to evaluate and compare the performance of different models, and evaluate the significance of any differences observed.

The rest of the chapter is structured as follows: section 3.1 describes the tasks in general, and the specific datasets we use for them. Section 3.2 looks at the software we have written to conduct the experiments. Specifically, section 3.2.1 covers the architecture, while section 3.2.2 describes the implementation. In section 3.3 we briefly describe the computing resources used for running the experiments, while in section 3.4 we look at how we evaluate performance. Finally, in section 3.5 we summarise the chapter.

3.1 The Tasks

All of the experiments in this thesis are undertaken using two sequence labelling tasks, with a separate dataset for each task. The tasks are **named entity recognition (NER)**

and **part-of-speech tagging (POS tagging)**. We chose these tasks because they are well-known to the NLP community, with standard datasets available for comparison with other approaches in the literature. In addition, the tasks contrast with each other in the type of problem they pose: POS tagging involves labelling single words in a sentence whereas NER involves identifying and labelling entities that could range across several words. In this section we briefly summarise the tasks and the datasets we use for them.

3.1.1 Named Entity Recognition

Named entity recognition involves the identification of the location and type of a set of pre-defined entities within a text. For example, within a bioinformatics domain the entities might be proteins, cell compartments or phases, whereas in an astronomy domain the entities might be planets, stars and other stellar objects. NER is often used as the first stage in a larger process. Examples include systems for information extraction, question answering and statistical machine translation. Borthwick (1999) gives a comprehensive overview of NER for English.

NER emerged as a sub-task of information extraction from a series of meetings known as the Message Understanding Conferences (MUC) (Grishman and Sundheim, 1996). These conferences were originally designed to encourage and foster research on automated analysis of messages, primarily with a view to military applications. NER was developed as a sub-task in the sixth conference (MUC-6) in 1995. Since then it has been established as a task in its own right, and is frequently the subject of competitions and challenges held at conferences and workshops. Examples include the Conference on Computational Natural Language Learning (CoNLL) Shared Tasks (Kim Sang and Meulder, 2003) and the Information Retrieval and Extraction Exercise (IREX) (Sekine and Isahara, 1999). NER was originally applied to English, but has since been used with other languages, including German (Kim Sang and Meulder, 2003), Dutch, Spanish (Kim Sang, 2002) and Japanese (Sekine and Isahara, 1999). NER systems can generally be classified into two broad categories: **rule-based** systems (Farmakiotou et al., 2000; Kim and Woodland, 2000) and **statistical** methods. In recent years, statistical approaches have become more prevalent and encompass a wide range of models, including decision trees (Sekine et al., 1998), hidden Markov models (Klein et al., 2003), maximum entropy models (Klein et al., 2003; Curran and Clark, 2003), support vector machines (SVMs) (Asahara and Matsumoto, 2003), voted perceptrons (Collins,

2002) and CRFs (McCallum and Li, 2003).

As an illustration of the annotation scheme for NER, consider the following sentence:

```
Nato declines comment on fighting in Iraq.
```

When annotated with labels representing entities relevant to the domain, the sentence becomes:

```
Nato/I-ORG declines/O comment/O on/O fighting/O in/O Iraq/I-LOC ./O
```

We see that each token (including the full-stop) is annotated with an entity label. The symbol `O` denotes the fact that a token is not part of a specific entity (i.e. it is outside an entity). The label `I-ORG` denotes the fact that a token is a *organisation*, while the label `I-LOC` denotes the fact that a token is a *location*. In general, a label of the form `I-X` signifies that a token is part of an entity of type `X` , and all words in an annotated dataset are given either a label of this form or an `O` label.[1]

For our experiments we use the CoNLL-2003 shared task dataset for English (Kim Sang and Meulder, 2003). This dataset consists of three sections: a training set, development set and test set. The size of these sets, in terms of number of sentences and tokens, is shown in Table 3.1. The dataset was compiled from Reuters news stories. The training and development sets are comprised of ten days' worth of news coverage from August 1996, while the test set consists of articles from December 1996. For this dataset there are four entities: persons (PER), locations (LOC), organisations (ORG) and miscellaneous (MISC). With this dataset, as with NER in general, the entities are fairly sparse and the vast majority of words have the `O` label. This means that NER is really the task of spotting the occasional entity against a background of non-entities. In the CoNLL-2003 dataset the words were POS-tagged automatically using the memory-based MBT tagger (Daelemans et al., 2002).

The CoNLL-2003 dataset was designed with the NER task in mind and allows us to benchmark our results against those obtained by a number of other models that were employed in the shared task. The best performing system on the task attained an F score of 93.87% on the development set and 88.76% on the test set. This was a classifier combination framework involving a linear classifier, a maximum entropy

[1] Here we are using an annotation format where there is an alternative label `B-X` for an entity of type `X`. This label is used for the first token of an entity of type `X` when the previous token was part of a different entity also of type `X`. Except for this case, tokens that are part of entities of type `X` are labelled with `I-X`. This is known as the IOB annotation scheme (Ramshaw and Marcus, 1995).

Objects	Training	Development	Test
Tokens	203,621	51,362	46,435
Sentences	14,987	3,466	3,684

Table 3.1: Numbers of tokens and sentences in the CoNLL-2003 shared task dataset.

model, a transformation-based learning model and an HMM (Florian et al., 2003). Other high scoring systems included a maximum entropy model (Chieu and Ng, 2003) and a combination of a maximum entropy model and an HMM (Klein et al., 2003).

3.1.2 Part-of-Speech Tagging

POS tagging involves labelling each word in a sentence with its part-of-speech. A word's part-of-speech indicates its syntactic function, such as noun, verb or adjective, as well as other information like number and tense. POS tagging has a long history, dating back to the mid-1960s. The first well-known tagger that assigned tags to words on the basis of local syntactic information, as opposed to just looking up tags in a dictionary, was that of Klein and Simmons (1963). The first probabilistic tagger was probably that of Stolz et al. (1965), which used conditional probabilities calculated from tag sequences to assign tags to words after some pre-processing steps. Initially, POS tagging was focused on English but over the years it has been applied to many other languages. As with NER, most POS tagging systems can be roughly classified as either rule-based or statistical. The best-known rule-based approaches are probably those of Brill (1995). As for the statistical approaches, many models have been proposed. These include hidden Markov models (Kupiec, 1992; Merialdo, 1994; Jelinek, 1985), decision trees (Schmid, 1994), neural networks (Benello et al., 1989), memory-based learning models (Daelemanns et al., 1996) and maximum entropy models (Ratnaparkhi, 1996).

To illustrate the annotation scheme for part-of-speech tagging, let us refer back to our example sentence from the previous section. Annotated with POS tags, this would look like:

```
Nato/NNP declines/VBZ comment/NN on/IN fighting/VBG in/IN Iraq/NNP ./.
```

In this case, then, `VBZ` denotes a verb, `NN` denotes a noun, etc. In an annotated dataset every word is given such a label.

For our POS tagging experiments we use an amended version of the CoNLL-2000 shared task dataset for English (Kim Sang and Buchholz, 2000). This dataset was extracted from the Penn Treebank (Marcus et al., 1993) for use in the shared task. In addition, it was relabelled with POS tags using the Brill tagger (Brill, 1994). For our experiments, however, we have restored the original, hand-annotated POS tags that appear in the Penn Treebank.

In Chapter 1 we mentioned scaling as a current limitation of CRFs. In general, CRFs do not scale well to complex structured labelling tasks, where the resulting graphical model is densely connected. In these cases inference can be intractable. The situation is simpler in the case of a (first order) linear chain CRF. As we saw in Chapter 2, for linear chain CRFs there are efficient dynamic programming algorithms to handle inference. However, even linear chain CRFs do not scale well to tasks with a large number of labels. In general, the time complexity for inference in a linear chain CRF scales roughly with the square of the number of labels in the dataset. Therefore, for sequence labelling tasks with a large number of labels, inference can be intractable unless some explicit attempt is made to scale the model. An example of such a task is POS tagging, where the label set size is often in the range of 40 - 45 tags. In his thesis, Cohn (2006) examines different approaches to scaling CRFs to efficiently handle such tasks. The work in that thesis was developed in parallel to the work presented here. Consequently, we address the scaling problem for POS tagging using a different approach: we collapse the label set. The shared task dataset contains 45 different POS tags. We collapse the number of tags to 5, employing the procedure used by McCallum et al. (2003). The procedure is as follows:

- All types of noun are collapsed to a single category **N**.
- All types of verb are collapsed to a single category **V**.
- All types of adjective are collapsed to a single category **J**.
- All types of adverb are collapsed to a single category **R**.
- All other POS tags are collapsed to a single category **O**.

The dataset provided with the shared task only includes a training set and a single development/test set. In order to provide a separate development set, while leaving the test set untouched, we randomly partition the training set into a training subset and a development subset, using the same ratio of sizes as for the NER dataset. As a result, we obtain training, development and test sets with sizes given in Table 3.2.

Objects	Training	Development	Test
Tokens	172,898	38,829	47,377
Sentences	7,300	1,636	2,012

Table 3.2: Numbers of tokens and sentences in the CoNLL-2000 shared task dataset.

3.2 Software

In this section we describe the software that we have written to undertake the experiments for the thesis. We first describe the general architecture at an abstract level, then give details as to how the architecture is implemented in the software itself.

3.2.1 Architecture

Figure 3.1 shows the software architecture for the experimental pipeline. In general, to run a single experiment we train a model using a training dataset, then use the trained model to decode other datasets, such a development set and/or a test set. Therefore, the two primary stages in the experimental pipeline are **training** and **decoding**. Before being able to train the model, however, we must define the features to be used in the model, and ascertain where those features are active in the datasets. We therefore have **feature instantiation** (or definition) and **feature extraction** stages as pre-processing steps to the training stage. Consequently, our experimental pipeline consists of four stages, and these are represented by the rectangular boxes in the figure. The elliptical objects in the figure represent the inputs and outputs of each stage. In reality these elliptical objects take the form of text files.

3.2.1.1 Feature Instantiation

In Chapter 2 we defined a feature function as being a conjunction of predicates specified on a particular **clique** of a sequence. We now imagine a particular feature, f_{103}, that could be used in a POS tagging task. Specified on clique t of a sequence with observations $\mathbf{o}$ and on a clique with POS labels s' and s (with s being thought of as occupying the current position and s' the previous), this feature is defined as:

$$f_{103}(s', s, \mathbf{o}, t) = \begin{cases} 1 & \text{if current\_word}(\mathbf{o}, t) = \text{“the” holds and } s = \text{“DET”} \\ 0 & \text{otherwise} \end{cases}$$

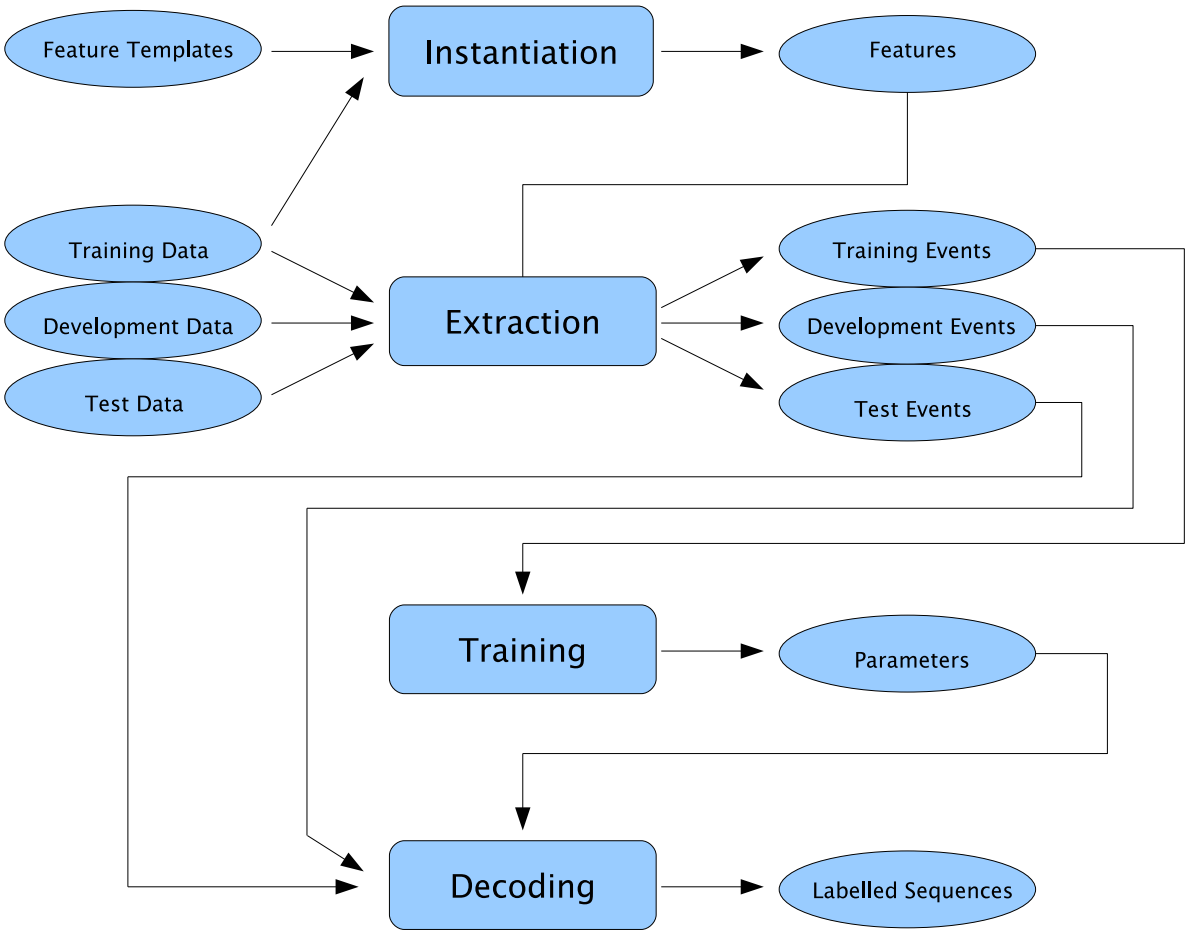

Figure 3.1: Software architecture for the experimental pipeline.

So this feature is **active** on any clique for a given labelling of the observations if and only if the current word is “the” and the current POS label is “DET”. In fact, in this case the feature has no predicates that test s'. Note that the feature here, and indeed all the features used in our experiments, are binary-valued. However this not a requirement, and in principle features could be real-valued. In order to define a model we must decide upon a set of features like f_{103} to include in the model. We do this by defining a set of **feature templates**. These are like blue-prints for creating features such as f_{103} above, and are motivated by our linguistic intuition about the properties of a sequence that are important if determining its labelling. For example, f_{103} could have been created, or **instantiated**, from a feature template such as:

$$FT_{X,Y}(s', s, \mathbf{o}, t) = \begin{cases} 1 & \text{if current\_word}(\mathbf{o}, t) = X \text{ holds and } s = Y \\ 0 & \text{otherwise} \end{cases}$$

Having defined a set of feature templates, we instantiate features from the templates by passing through the training data and collapsing the non-grounded forms for the predicates in each template on each clique of each sequence to the grounded forms defined at the clique. For example, f_{103} would have been instantiated from $FT_{X,Y}$ by instantiating X to "the" and Y to "DET".

For the experiments we carry out in this thesis, we only ever use features that have been created in this way. These are called **supported** (or sometimes **attested**) features because they have all been seen at least once in the training data. It is possible to include other, non-supported features in the model – features that have not been observed in the training data. One way to do this would be to instantiate all possible grounded conjunctions of predicates from the feature templates, rather than just that subset that have been observed. As noted by Sha and Pereira (2003), including these unsupported features in a model may in principle cause it to perform better because the new features can attach negative weights to transitions that should be discouraged if a certain observation predicate is active. However, including these features does require some form of regularisation to be applied during training because without this the parameters associated with the unsupported features will tend to negative infinity. Some of the results we present later refer specifically to the use of unregularised models in a LOP. In order to compare these with the corresponding unregularised models on a consistent set of features, we chose not to include unsupported features. However, in principle, our findings should still hold for LOPs with models including non-supported features.

Typically the feature templates naturally fall into different categories based upon the dependencies that the features instantiated from them are intended to model. For example, some templates may instantiate to features that model the dependency of a word's NER label on the word's POS label, while others may instantiate to features that model dependencies between the NER label and the word's orthographic properties. The specific feature templates that we use for our experiments are described in the next chapter, when we introduce the reference models.

3.2.1.2 Feature Extraction

Having defined a set of features to be used in the model, we need to determine where each feature is active in the dataset. In Chapter 2 we saw that during each iteration of CRF training we must evaluate the expected value of each feature under the current model. This involves a weighted sum over all possible labellings. As a result, we must determine which features are active in the training data for all possible labellings of the sequences, not just the observed one (the gold-standard). Additionally, when we use a trained model to decode a development set or test set (as we also saw in Chapter 2), we use a Viterbi algorithm which requires us to know which features are active on all cliques of the dataset for all possible labellings – with different possible labellings corresponding to different paths through the Viterbi lattice. Therefore, the feature extraction stage in the experimental pipeline determines the active features for all labellings of all the datasets we are using: training set, development set and test set. This is represented in Figure 3.1 by the three arrows going into, and out of, the extraction stage box.

3.2.1.3 Training

The output of the extraction stage, giving information regarding feature activations, is the single input we need for the training stage. During the training stage the model is fitted iteratively using the procedures described in Chapter 2. The fitting will cease when some termination condition is satisfied. Typically this is a threshold for the minimum absolute or relative change in the objective function, but could also be a maximum number of iterations for example. The output of the fitting stage is a set of parameters representing the converged model. We often also output other sets of parameters, for example parameters representing the state of the model at different iterations. This allows for analysis of the learning process of the model during training.

3.2.1.4 Decoding

Having arrived at a trained model we may then use it to decode another dataset, typically a development set or test set. Our inputs to the decoding stage are therefore the parameter set (output from the training stage) and the feature activation information for the dataset to be decoded (output from the extraction stage).

3.2.2 Implementation

The software to implement the four stages in the experimental pipeline is written in a combination of C++ and Perl. In general, Perl scripts are used to control high level behaviour, while C++ code handles lower-level processing. The C++ code therefore executes the functionality associated with the four stages of feature instantiation, feature extraction, model training and model decoding. In fact, all the stages are implemented by a single C++ binary. By contrast, Perl scripts define the configuration settings associated with the four stages such as input and output file names, and convergence conditions for the training. For some experiments, the experimental pipeline described above is executed many times over with a different value for a hyperparameter each time. In this case the Perl scripts define the range of hyperparameters to be used.

As we discussed in Chapter 2, during training we optimise an objective function, typically a penalised log-likelihood of the training data. Because an optimal set of parameters cannot be found analytically, we must resort to using iterative numerical routines. In particular, we use the **limited memory variable metric (LMVM)** routine (Nocedal, 1980), and for a small number of experiments in Chapter 5 we use a variant on this procedure called **bounded limited memory variable metric (BLMVM)** (Benson and More, 2001). Both routines are implemented in the Toolkit for Advanced Optimisation (TAO) libraries (Benson et al., 2005). We also use the Portable Extensible Toolkit for Scientific Computation (PETSc) (Balay et al., 2004), which provides a set of vector structures that interface easily with TAO. Underneath both TAO and PETSc sits a Message Passing Interface (MPI)[2] communication layer, which allows for efficient communication over a network when running parallelised experiments. We use two different implementations of MPI, called MPICH[3] and LAMMPI[4]. Most of the experiments reported in this thesis involve only single-node processing, but the **co-operative training** framework described in Chapter 7 requires a parallelised architecture and this makes explicit use of the MPI layer. In fact, the architecture for the co-operative training software is a little more complex than the architecture here, so we describe it separately in that chapter.

[2] http://www-unix.mcs.anl.gov/mpi
[3] http://www-unix.mcs.anl.gov/mpi/mpich
[4] http://www.lam-mpi.org

3.3 Computing Resources

The experiments presented in this thesis were all conducted using the computing resources within the School of Informatics at the University of Edinburgh. These resources broadly fall into two categories:

- **compute servers**: a small number of standalone machines powerful enough to run an experiment in isolation. These machines are each equipped with four Intel Xeon 2.8GHz processors and 4 or 5GB of RAM.
- **clusters**: four Beowulf clusters of machines. The number of nodes in a cluster and the specification of the nodes, varies considerably between the clusters. The newest cluster, which was used the most for our experiments, consists of 34 nodes each equipped with four Intel Pentium 4 3.0GHz processors and 4GB of RAM.

3.4 Performance Evaluation

3.4.1 Performance Scores

In our experiments we must measure the performance of different models in order to be able to compare them. We do this using standard measures employed by other researchers. For the POS tagging experiments we measure **per-token accuracy** (shortened to **accuracy** for the rest of the thesis). This is simply the proportion of tokens (words) that are labelled correctly. NER, however, is a little more complex as it involves chunks, or **entities**. We want a measure that encodes how well we have correctly identified entity boundaries, and a simple measure such as accuracy will not represent this clearly. We therefore use **F score**, which is turn is derived from **precision** and **recall**. Precision (*P*) and recall (*R*) are defined as:

$$P = \frac{TP}{TP+FP} \tag{3.1}$$

$$R = \frac{TP}{TP+FN} \tag{3.2}$$

where TP is the number of **true positives** (correctly identified entities), FP is the number of **false positives** (incorrectly identified entities) and FN is the number of false negatives (incorrectly omitted entities). The statistics TP, FP and FN may relate

to a particular entity, or be global, across all entities. As they are all non-negative, it is clear that precision and recall both take values in the interval from 0 to 1. Ideally, we would like achieve a high value (i.e. close to 1) for both precision and recall. Indeed, if we correctly label every word, we obtain a score of 1 for both precision and recall. In general, however, trying to maximise recall may lead to over-prediction of a particular chunk, which in turn leads to low precision. Conversely, trying to maximise precision may lead to a under-prediction of a particular chunk, which in turns leads to low recall. We therefore require a measure that encodes both precision and recall. The standard candidate is **F score**, which is defined by:

$$F_\alpha = \frac{1}{\alpha\frac{1}{p} + (1-\alpha)\frac{1}{R}} \tag{3.3}$$

The parameter α determines the weighting of precision and recall. Usually it is set to 0.5, with the F score becoming:

$$F_{0.5} = \frac{2PR}{P+R} \tag{3.4}$$

It is this definition (3.4) that we use as the performance measure in our NER experiments.

3.4.2 Significance Testing

When assessing the difference in the performance of two models using the measures described above, we must look not just at the absolute difference in the values obtained, but also the statistical significance of that difference. To do this we need an appropriate statistical test for sequence labelling problems. We use a test popularised by Sha and Pereira (2003) called the **matched-pairs test** (Gillick and Cox, 1989).

The matched-pairs test assumes that the stream of labelling decisions (i.e. sentences labelled by a CRF in our case) can be divided into units such that, for a given model, the errors that the model makes on a given segment are independent of the errors that it makes on any of the other segments. With a CRF on the sequence labelling tasks we consider in this thesis, it is reasonable to take a sentence as such a unit. The test compares two models by measuring the difference in the number of errors (per-token labelling errors) each model makes on each sentence. Let N_α^i be the number of errors made by model α on sentence i, where i runs over all N sentences in the

dataset. Then $Z_i = N_1^i - N_2^i$ measures the difference in the number of errors made by model 1 and model 2 on sentence i. Suppose we want to know the true mean difference in the number of errors made by the models across all possible sequences that could be encountered. A natural estimate for this mean difference, μ_z, is the sample mean $\hat{\mu}_z = \frac{1}{N}\sum_{i=1}^{N} Z_i$. In addition, a natural estimate for the variance of this difference is the sample variance $\hat{\sigma}_z^2 = \frac{1}{N-1}\sum_{i=1}^{N}(Z_i - \hat{\mu}_z)^2$. Now, if we define W by:

$$W = \frac{\hat{\mu}_z}{\hat{\sigma}_z/\sqrt{N}} \tag{3.5}$$

then for large N (e.g. larger than 50) W will have a distribution that is approximately normal, with unit variance. If we assume that on average the two models perform equally well from the point of view of per-token labelling errors, then μ_z (and W for infinitely large N) would be zero. Therefore, if we want to test whether two models have significantly different accuracies, we form a null hypothesis $\mathbf{H_0} : \mu_z = 0$ and test it by seeing how far the estimate W is from 0. We reject the hypothesis if W is sufficiently far from zero that under $\mathbf{H_0}$ W would have taken the given value (or a value further from 0) with a probability below some threshold that we define. As this corresponds to significant *difference* in accuracies at the threshold in question, the test is a *two-sided* one. However, if we instead want to establish, for example, whether the first model has an accuracy that is significantly *higher* than the second model, we form an alternative hypothesis $\mathbf{H_1} : \mu_z < 0$. We reject the hypothesis if W is sufficiently greater than zero that under $\mathbf{H_1}$ W would have taken the given value (or a higher value) with a probability below the threshold. This is a *one-sided* version of the test. In our experiments we use the one-sided version of the test and typically set the threshold at 5% ($p < 0.05$), but we state specific thresholds whenever significance tests are used later in the thesis.

By definition this test is based inherently on accuracy rather than any entity-oriented measure such a F score, so by using it we are measuring statistical significance of differences in accuracy, not F score. This could potentially pose a problem for comparison of models on NER. However, experience has shown that model performance rankings based on accuracy and F score are almost always the same. Hence using this accuracy-based statistical test is a reasonable substitute for an alternative entity-based one. It is possible to measure the statistical significance of differences in F scores directly using bootstrapping approaches (Yeh, 2000). However, Sha and Pereira (2003) found that such measures are often swamped by variance, and recommend the simpler matched

pairs test instead. We use the matched-pairs test in the thesis for both POS tagging and NER.

3.5 Summary

In this chapter we have presented the experimental framework that we use to conduct the experiments for this thesis. In particular, we have described:

- The two labelling tasks, named-entity recognition and part-of-speech tagging, that we use to illustrate our ideas throughout the core chapters.
- The experimental pipeline and software architecture, including the stages covering feature instantiation, feature extraction, and training and decoding of models.
- The measures we use to evaluate and compare the performance of different models, and evaluate the significance of any differences observed.

Having presented the general experimental setup in this chapter, we move on in the next chapter to present results for the baseline models and to empirically demonstrate the CRF's tendency to overfit.

Chapter 4

Reference Models and Overfitting

In this chapter we describe the standard CRF models that we use throughout the rest of the thesis, and build upon in later chapters. We refer to these as **reference models**. The basic performance results that we provide for these models will be used as baselines for comparison later. Also in this chapter we demonstrate the tendency for a CRF to overfit the data during training, an idea that we introduced and discussed in Chapter 2.

4.1 The Reference Models

Throughout the thesis, for both NER and POS tagging, we use some reference CRF models. These models are called SIMPLE and STANDARD. They differ in the features that they contain. Each model has two versions, one for each task. The feature templates used to generate the SIMPLE model are the same for both NER and POS tagging. For the STANDARD model, however, the feature templates vary with task. Table 4.1 shows the templates used to generate the SIMPLE model on both tasks. Note that s_t denotes the label at position t in the sequence, and w_t denotes the word at position t. The *label* in this context depends on the task. For POS tagging the label is

Label predicates	Observation predicates
$s_{t-1} = s', s_t = s$	
$s_{t-1} = s', s_t = s$	$w_t = w$
$s_t = s$	$w_t = w$

Table 4.1: Feature templates generating the SIMPLE model.

Label predicates	Observation predicates
$s_{t-1} = s', s_t = s$	
$s_{t-1} = s', s_t = s$	$w_t = w$
$s_t = s$	$w_{t-2} = w$
	$w_{t-1} = w$
	$w_t = w$
	$w_{t+1} = w$
	$w_{t+2} = w$
	$w_{t-2} = w', w_{t-1} = w$
	$w_{t+1} = w', w_{t+2} = w$
	$p_t = p$
	$p_{t-2} = p', p_{t-1} = p$
	$p_{t+1} = p', p_{t+2} = p$

Table 4.2: Feature templates generating *n*-gram features.

the POS tag, whereas for NER the label is the NER label. The feature templates in Table 4.1 represent basic structural dependencies between consecutive labels, and the label and word at a given position. When instantiated, these feature templates generate $24,819$ SIMPLE model features for NER and $18,482$ SIMPLE model features for POS tagging.

The feature templates used to generate the STANDARD model for each task are a superset of those generating the SIMPLE model. They may be subdivided into three categories according to the type of features they generate:

1. **N-gram features**. These are features that involve predicates defined on the observations that are *n*-grams of words, and, for NER, POS tags. Table 4.2 shows the feature templates that are used to generate these features. The templates used to generate the STANDARD model for POS tagging are only those residing above the dividing line. By contrast, for the NER STANDARD model all templates in Table 4.2 are used. In the case of NER, feature templates treat POS tags as observations. These are the ones involving symbols such as p_t, which denotes the POS tag at position t in the table.

   To clarify the information contained in the table, let us take an example. The fifth line in the table shows a feature template (called FT_5 here for illustration)

Label predicates	Observation predicates
$s_t = s$	w_t contains a digit
	w_t contains an upper case character
	w_t contains a hyphen
	w_t contains a period
	w_t contains punctuation
	w_t is all digits
	w_t is a number
	w_t is alphanumeric
	w_t is only Roman numerals
	w_t is all uppercase
	w_t is all lowercase
	w_t is of mixed case
	w_t is a title
	w_t is an initial
	w_t is an acronym
	w_t has prefix x of length l, $l = 1 \ldots 4$
	w_t has suffix x of length l, $l = 1 \ldots 4$
	w_t has length x

Table 4.3: Feature templates generating orthographic features.

Label predicates	Observation predicates
$s_t = s$	w_{t-1} has Collins' format f
	w_t has Collins' format f
	w_{t+1} has Collins' format f
	w_{t-2}, w_{t-1} have Collins' format f, f'
	w_{t-1}, w_t have Collins' format f, f'
	w_{t+1}, w_{t+2} have Collins' format f, f'
	w_{t-2}, w_{t-1}, w_t have Collins' format f, f', f''
	w_{t-1}, w_t, w_{t+1} have Collins' format f, f', f''
	w_t, w_{t+1}, w_{t+2} have Collins' format f, f', f''

Table 4.4: Feature templates generating Collins' features.

which is defined by:

$$FT_5 = \begin{cases} 1 & \text{if current\_word} = w \text{ and current\_label} = s \\ 0 & \text{otherwise} \end{cases}$$

The *n*-grams shown in Table 4.2 are based on those described by Curran and Clark (2003), where a similar set was shown to be effective in a standard maximum entropy model for NER. The feature templates in the table consist only of unigrams and bigrams of words and POS tags. By including higher order predicates (trigrams and higher) we risk overfitting. Indeed, we use such features in the next section to demonstrate the overfitting effect.

2. **Orthographic features**. In addition to the *n*-gram features, the STANDARD model for NER contains **orthographic** and **Collins'** features. Feature templates that generate orthographic features are shown in Table 4.3. These templates consist of predicates that pick out some orthographic property of the current word, such as whether it is capitalised, whether it contains a digit, etc.

3. **Collins' features**. The term *Collins' feature* is our terminology, and refers to features that contain a special predicate defined by Collins (2002). The predicate maps words to **word classes**, where a word class consists of words with the same orthographic properties. Specifically, each character in a word is mapped to a symbol and adjacent characters with the same symbol are then merged together. For example, the word `Hello` would map to `Aa`, the initials `A.B.C.` would map to `A.A.A.` and the number `1,234.567` would map to `0,0.0`. Table 4.4 shows the feature templates that are used to generate Collins' features. As can be seen from the table, the templates involve *n*-grams of words (in a window around the current word) over which the Collins' predicate is applied.

When instantiated, these feature templates generate 450,346 features for the STANDARD model for NER and 189,339 features for the STANDARD model for POS tagging.

Table 4.5 gives F scores for the reference models on the NER task, for both development set and test set. Table 4.6 gives accuracies for the corresponding models on the POS tagging task. All the models are unregularised. Conventional regularisation of the STANDARD model (including use of a Gaussian prior) is discussed in Chapter 5. As

Model	Development	Test
SIMPLE	79.53	69.22
STANDARD	88.21	81.60

Table 4.5: F scores for reference models SIMPLE and STANDARD on NER.

Model	Development	Test
SIMPLE	95.15	94.84
STANDARD	96.76	96.32

Table 4.6: Accuracies for reference models SIMPLE and STANDARD on POS tagging.

expected from the feature sets described earlier, the STANDARD model significantly outperforms the SIMPLE model at a significance level of $p < 0.05$.

Table 4.7 gives representative times for training and decoding of the STANDARD and SIMPLE models on NER. The corresponding times on POS tagging show similar trends but are in general a little lower because the POS tagging models have smaller feature sets, having been instantiated on a smaller training dataset.

4.2 Overfitting

In Chapter 2 we made the claim that CRFs have a tendency to overfit small datasets that are used to train them. To demonstrate this, we take the NER STANDARD model as a base and define a sequence of models via sets of additional features. These sets form a sequence of supersets. We then train each model in the sequence and compare results to the STANDARD model. Table 4.8 shows the feature templates from which the additional sets of features are generated, in a sequence $A_1 \ldots A_4$. These involve other *n*-grams of words and POS tags not used in the STANDARD model, including higher order *n*-grams. Note that the sets are increasing supersets in the sense that model A_2, for example, includes all the feature templates in model A_1 as well as the additional ones from the rows labelled A_2 in the table.

As we add more features to the STANDARD model, there are potentially two conflicting effects taking place. On the one hand we give each model greater modelling power, which could lead to increased performance. One the other hand, we risk an

Model	Dataset		
	Training	Development	Test
STANDARD	90 mins	50 secs	40 secs
SIMPLE	25 mins	25 secs	20 secs

Table 4.7: Representative times for training and decoding of the STANDARD and SIMPLE models on NER.

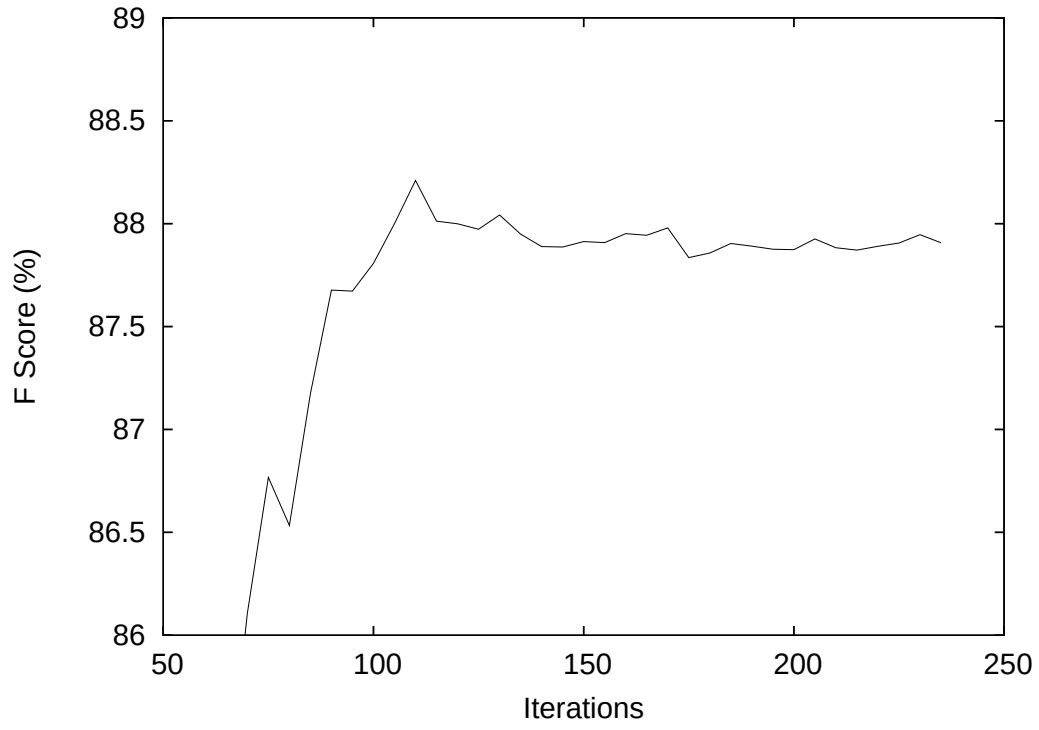

Figure 4.1: Development set performance during training for the STANDARD model on NER.

increased effect of overfitting as the extra degrees of freedom introduced through the new features fit to the idiosyncrasies of the training data itself rather than the systematic properties of the underlying distribution. Table 4.9 gives F scores for the models $A_1 \ldots A_4$. The table shows both converged scores and best scores, where *best* refers to the model that obtains the highest score on the development set during training. We see from the table than the model A_1 performs roughly equally well as the STANDARD model (the converged F score is slightly higher, although the best F score is a little lower) but as we continue to add more features we obtain models that perform increas-

Label predicates	Observation predicates	Model
$s_t = s$	$p_{t-2} = p$	A_1
	$p_{t-1} = p$	
	$p_t = p$	
	$p_{t+1} = p$	
	$p_{t+2} = p$	
	$w_{t-1} = w'$, $w_t = w$	A_2
	$w_t = w'$, $w_{t+1} = w$	
	$p_{t-1} = p'$, $p_t = p$	A_3
	$p_t = p'$, $p_{t+1} = p$	
	$w_{t-2} = w''$, $w_{t-1} = w'$, $w_t = w$	
	$w_{t-1} = w''$, $w_t = w'$, $w_{t+1} = w$	
	$w_t = w''$, $w_{t+1} = w'$, $w_{t+2} = w$	
	$p_{t-2} = w''$, $p_{t-1} = w'$, $p_t = w$	A_4
	$p_{t-1} = w''$, $p_t = w'$, $p_{t+1} = w$	
	$p_t = w''$, $p_{t+1} = w'$, $p_{t+2} = w$	

Table 4.8: Additional sets of feature templates added to the STANDARD model templates to demonstrate overfitting.

Model	Best	Conv
STANDARD	88.21	87.91
A_1	88.03	88.00
A_2	87.83	87.70
A_3	87.34	87.34
A_4	87.10	86.89

Table 4.9: Development set F scores for feature template sequence models.

ingly badly as overfitting increases. Clearly, we can see that the tendency to overfit increases steadily as the number of features is increased. As a result, we conclude that to apply CRFs effectively here, application of some form of regularisation is vital.

Table 4.9 we shows that the STANDARD model overfits the training data least. However, even with the STANDARD model, there may still be a certain degree of overfitting taking place. Indeed this may be observed by considering the path that is taken through the model space as the STANDARD model is trained. As the training continues and the model is fitted ever more closely to the training data distribution, there comes a point at which the model's performance on the development set starts to diminish. This is illustrated in Figure 4.1, which shows the NER STANDARD model's performance on the development set during training. The best model with respect to the development set, and the one for which we may expect the best generalisation behaviour, lies somewhere in the region between iterations 100 and 120. Beyond this point, the model starts to fit more and more closely to the details of the training data distribution rather than the underlying distribution.

Note that with the examples in this section we are demonstrating situations in which a CRF overfits. Previous work with CRFs on various NLP tasks has also demonstrated the tendency for CRFs to overfit, and this has led to the need for some form of regularisation when applying CRFs to such tasks. However, we are not claiming that CRFs *always* overfit. In situations where increasingly large amounts of data are available, in relation to model complexity, CRFs will overfit less and less.

4.3 Summary

In this chapter we have described the reference models that we use, and build upon, in later chapters. We have presented the performance results for these models on both NER and POS tagging. In addition, we have demonstrated the tendency of a CRF to overfit the training data by creating a sequence of models with increasing numbers of features. Having shown the existence of the overfitting problem, and the consequent requirement for some form of regularisation when training CRFs, in the next chapter we investigate conventional approaches to CRF regularisation, and propose some extensions to them.

Chapter 5

Conventional Regularisation for CRFs

In Chapter 2 we explained the general phenomenon of a model overfitting the data on which it is trained, and described some of its symptoms. Then, in the last chapter, we looked at the specific case of overfitting by CRFs. Using NER as an example, we saw how a CRF may overfit a dataset in some circumstances. We did this by training a sequence of CRFs with increasingly large feature sets. Much CRF research to date has supported the view that CRFs have a tendency to overfit, particularly when they are applied to real-world tasks. Examples of such work include application of CRFs to tasks such as named entity recognition (McCallum and Li, 2003), noun-phrase chunking (Sha and Pereira, 2003) and information extraction from research papers (Peng and McCallum, 2004). We may conclude, therefore, that successful application of CRFs requires some form of regularisation to address this tendency to overfit.

Standard approaches to regularising CRFs, and log-linear models in general, have focused on the use of a Gaussian prior distribution over the model parameters. The choice of a Gaussian distribution (as opposed to some other) is primarily due to the Gaussian being well understood and easy to implement. However, there are a number of questions that arise here. Firstly, it is not clear that the Gaussian distribution is the most natural choice for a prior all the time. With a wide variety of tasks and a correspondingly diverse range of label distributions, it seems likely that in some cases other prior distributions may be more appropriate. Secondly, a typical application of a Gaussian prior involves constraining the distribution variance to be fixed across all model parameters. This simplification aids computation, avoiding the need to search a hyperparameter space of large dimension in order to fit the prior. However, varying the value of the variance of a Gaussian prior distribution for a particular model parameter influences the degree of regularisation applied to that parameter. Therefore, by applying a

fixed variance across all parameters we are effectively applying an equal regularising effect to each one. However, there may be some circumstances where we want to regularise some parameters more than others. For example, if the feature associated with one parameter is seen much more frequently than that of another, we may conjecture that its parameter does not require regularising to the same degree as the other one. So how might we decide on the level of regularisation for different parameters? A third question relates to the Gaussian prior mean. In a typical implementation using a Gaussian prior, the mean is assumed to be zero. This is again primarily for simplicity and avoids the need to search a joint space of both mean and variance to fit the prior. However, fixing the Gaussian mean at zero penalises positive and negative parameter values equal in magnitude by the same amount, thereby discouraging movement away from zero symmetrically. Since we may expect some parameters to have a positive value while others are negative, it is not clear that a sensible choice for a mean is zero itself – it could be a small positive value for example. So, what is the optimal choice for the Gaussian mean? What happens when we allow the mean to move a away from zero?

In this chapter we address these questions. Starting in section 5.2 we explore the use of alternative priors and compare these to the Gaussian. In particular, we experiment with **Laplacian** and the **Hyperbolic** priors. In section 5.5 we then move on to look at **feature-dependent regularisation**, where different model parameters are regularised to different degrees. We investigate different ways of grouping features together for regularisation, including the extreme case where each feature is regularised[1] to a different level. Then, in section 5.6, we experiment with non-zero values for the Gaussian mean. To complete the picture, we also consider very simple approaches to regularisation in the form of a feature cutoff, whereby parameters corresponding to features which occur infrequently in the training data are removed from the model altogether. We look at this with and without regularisation of the remaining parameters (in sections 5.1 and 5.4 respectively). Finally, towards the end of the chapter, in section 5.7, we look at an alternative formulation for regularisation of a CRF, which we call the **inequality CRF**. This involves relaxing the feature expectation constraints we discussed in Chapter 2. We show how this model is very similar in principle to regularisation with a prior, and compare the two approaches.

[1]Note that sometimes we will use a phrase such as “regularise the features” to mean “regularise the parameters associated with the features”.

5.1 Feature Cutoff

Before considering different families of prior distribution, we look in this section at a very simple approach to reducing overfitting, called **feature cutoff** (Ratnaparkhi, 1998). Feature cutoff is based on the idea that when a model is parameterised using features that are redundant (or, more generally, not very useful) for modelling the distribution at hand, these features will tend to fit to arbitrary properties of the training data rather than modelling the systematic properties of the distribution observed. Thus the model is likely to overfit the training data. By definition features which have this property are likely to occur very infrequently. As a consequence, if we assume that all infrequent features may have this property to some degree, then overfitting may be reduced by removing infrequent features from the model. Feature cutoff therefore involves removing features whose frequency falls below some threshold. For our experiments we have chosen arbitrary thresholds of 1, 2 and 3.

Table 5.1 gives F scores on NER for CRFs with the different feature cutoff thresholds. Table 5.2 gives accuracies on POS tagging for the corresponding models. The STANDARD model is included in each case for comparison. We include development set scores for completion, but the more interesting results are those for the test set in each case. Note that the significance tests we use in this section, and throughout the chapter, are at the $p < 0.05$ level.

From both tables we see that as we increase the cutoff threshold, in general scores decrease. This trend is not surprising: as we remove more features from the model, we take away useful modelling capability provided by the features in question. This effect will have increasing influence and will at some point outweigh any beneficial effect obtained from feature removal. As for other trends in the results, the patterns vary with task. For POS tagging, the model with features only of frequency one removed has greater accuracy than the STANDARD model, on both development and test sets. However, the scores it obtains are not significantly higher. The other models on POS tagging obtain lower scores than the STANDARD model, although these are not significantly lower. By contrast, on NER *all* feature cutoff models significantly underperform the STANDARD model, with the exception of the model with features only of frequency one removed, on the development set.

Although the specific effect of feature cutoff is not consistent across the two tasks, the general point we observe is that feature cutoff is not an effective method of regularisation for CRFs. In general we do not obtain significantly better results using this

Frequencies Removed	Development	Test
STANDARD	88.21	81.60
1	87.72	80.58
1 and 2	86.54	80.36
1, 2 and 3	86.34	80.80

Table 5.1: F scores for feature cutoffs on NER.

Frequencies Removed	Development	Test
STANDARD	96.76	96.32
1	96.79	96.39
1 and 2	96.68	96.23
1, 2 and 3	96.64	96.18

Table 5.2: Accuracies for feature cutoffs on POS tagging.

procedure. It seems that sufficient useful information is contained in features of low frequency that their removal harms the modelling capability of the CRF. This harming effect outweighs any regularising effect that their removal may have in principle. As we will see later in Chapter 8, some features of low frequency are highly informative, and removing these from the model can be very damaging to the model's performance. It seems, then, that the feature cutoff approach is in a sense too coarse, unable to distinguish between useful low frequency features and others that might be a source of overfitting.

5.2 Conventional Priors

Most approaches to CRF regularisation have focused instead on the use of a Gaussian prior distribution over the model parameters. However, other choices of prior are possible. In this section we investigate whether the Gaussian represents a natural choice for a CRF for sequence labelling tasks. To do so we compare three families of priors: the **Gaussian**, the **Laplacian** and the **Hyperbolic**. These priors are broadly similar in that they increasingly penalise the value of a model parameter as it moves further from the centre of the distribution. However, the priors differ in the specific form of

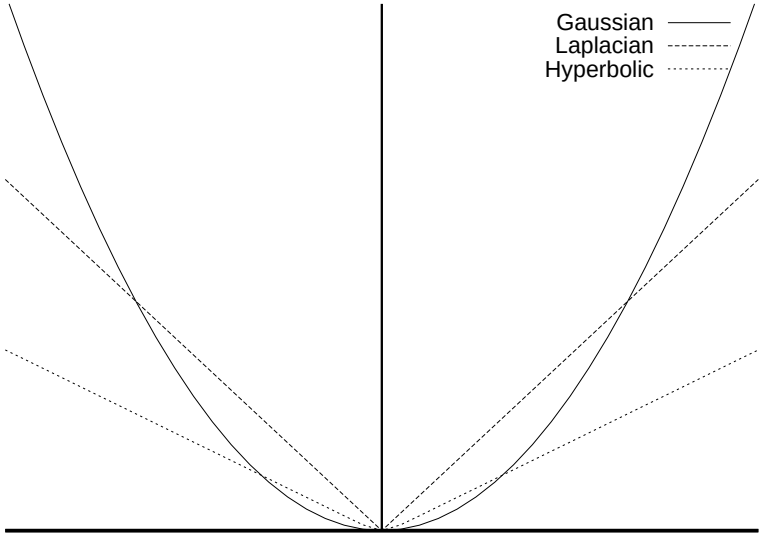

Figure 5.1: Qualitative depiction of penalty terms for the different priors.

the penalty term they employ. Figure 5.1 gives a qualitative depiction of the penalty term for each prior. As we explained in Chapter 2, this term appears in the objective function used to train the CRF.

5.2.1 Gaussian Prior

The most commonly used prior for CRF regularisation has been the Gaussian. Use of the Gaussian prior assumes that each model parameter is drawn independently from a **Gaussian** distribution, which is typically represented as:

$$p(\lambda_k) = \frac{1}{\left(2\pi\sigma_k^2\right)^{1/2}} \exp\left(-\frac{1}{2\sigma_k^2}\left(\lambda_k - \mu_k\right)^2\right) \tag{5.1}$$

The λ_k are the CRF model parameters, while the μ_k and σ_k^2 are **hyperparameters** of the Gaussian distribution. They represent the **mean** and **variance** of the distribution, respectively. Ignoring terms that do not affect the model parameters, the regularised log-likelihood with a Gaussian prior becomes:

$$LL(\lambda) - \frac{1}{2}\sum_k \left(\frac{\lambda_k - \mu_k}{\sigma_k}\right)^2 \tag{5.2}$$

We see that the penalty term for the Gaussian prior is a sum of terms of the form $\frac{1}{2}\left(\frac{\lambda_k-\mu_k}{\sigma_k}\right)$, one for each model parameter λ_k. At the optimal point a set of equations are satisfied. These equations, one for each λ_k, have the form:

$$E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] = \frac{\lambda_k - \mu_k}{\sigma_k^2} \tag{5.3}$$

We mentioned in the introduction to the chapter that the Gaussian prior is usually assumed to have zero mean. This involves setting $\mu_k = 0$ for each k. With respect to the Gaussian variances σ_k^2, there is a distinction between what we call **feature-independent regularisation** and **feature-dependent regularisation**. In the former, the parameters associated with each feature are regularised to an equal degree and so the corresponding variances σ_k^2 are constrained to be equal, i.e. $\sigma_k^2 = \sigma^2$ for all k. In the latter case, the variances σ_k^2 are allowed to vary with k. We present results for both these cases later in this chapter.

We can see from Equation 5.3 that use of a Gaussian prior enforces the constraint that the expected count of a feature under the model is *discounted* with respect to the count of that feature on the training data. This can be seen as somewhat similar in nature to the discounting schemes employed in language modelling (Chen and Rosenfeld, 2000). Under that analogy, use of the Gaussian prior corresponds to a form of **logarithmic discounting** in feature count space.

5.2.2 Laplacian Prior

Use of the Laplacian prior assumes that each model parameter is drawn independently from a **Laplace** distribution. This is typically represented as:

$$p(\lambda_k) = \frac{1}{2\beta_k} \exp\left(-\frac{|\lambda_k|}{\beta_k}\right) \tag{5.4}$$

The β_k are hyperparameters of the Laplace distribution, like the μ_k and σ_k for the Gaussian. Ignoring terms that do not affect the model parameters λ_k, the regularised log-likelihood with a Laplacian prior is given by:

$$LL(\lambda) - \sum_k \frac{|\lambda_k|}{\beta_k} \tag{5.5}$$

In this case we see that the penalty term for the Laplacian prior is a sum of terms of the form $\frac{|\lambda_k|}{\beta_k}$. At the optimal point the model satisfies:

$$E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] = \frac{\text{sign}(\lambda_k)}{\beta_k},\ \lambda_k \neq 0 \tag{5.6}$$

The numerator on the right-hand side in Equation 5.6 takes the value 1 if λ_k is positive and -1 if it is negative. Continuing our analogy with the language modelling world, we see from the form of the penalty term that use of the Laplacian prior is similar to applying **absolute discounting** in feature count space.

From Figure 5.1 we can observe that the derivative of the penalty term for the Laplacian prior with respect to a parameter λ_k is discontinuous at $\lambda_k = 0$. This leads to a potential problem in the evaluation of Equation 5.6 required by the optimisation libraries when training the model. To tackle this problem we use an approach described by Williams, who shows how the discontinuity may be handled algorithmically (Williams, 1995). His method leads to sparse solutions, where, at convergence, a substantial proportion of the model parameters can become zero. The result of this pruning effect is different, however, to feature induction, where features are included in the model based on their effect on log-likelihood.

5.2.3 Hyperbolic Prior

Use of the Hyperbolic prior assumes that each model parameter is drawn independently from the **Hyperbolic** distribution. Under this distribution, ignoring terms that do not affect the model parameters λ_k, the regularised log-likelihood is given by:

$$LL(\lambda) - \sum_k \log\left(\frac{\mathrm{e}^{\beta_k\lambda_k} + \mathrm{e}^{-\beta_k\lambda_k}}{2}\right) \tag{5.7}$$

The β_k are hyperparameters of the Hyperbolic distribution, as with the Laplacian. In this case we see that the penalty term for the Hyperbolic prior is a sum of terms of the form $\log\left(\frac{e^{\beta_k\lambda_k}+e^{-\beta_k\lambda_k}}{2}\right)$. At the optimal point the model satisfies:

$$E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] = \beta_k\left(\frac{\mathrm{e}^{\beta_k\lambda_k} - \mathrm{e}^{-\beta_k\lambda_k}}{\mathrm{e}^{\beta_k\lambda_k} + \mathrm{e}^{-\beta_k\lambda_k}}\right) \tag{5.8}$$

Note that, unlike the Gaussian and Laplacian priors, the penalty term for the Hyperbolic prior does not correspond directly to any form of discounting in the language modelling setting.

Prior	Development	Test
STANDARD	88.21	81.60
Gaussian	89.86	83.97
Laplacian	89.57	83.62
Hyperbolic	89.41	83.47

Table 5.3: F scores for different prior families on NER.

Prior	Development	Test
STANDARD	96.76	96.32
Gaussian	97.07	96.69
Laplacian	97.07	96.77
Hyperbolic	96.85	96.41

Table 5.4: Accuracies for different prior families on POS tagging.

5.3 Feature-Independent Regularisation

Having described the three prior families[2] in general in the last section, we now present results of experiments conducted using feature-independent regularisation with these priors. To recap, with feature-independent regularisation we fix the value of the adjustable regularising hyperparameter across all model parameters. We set this single value using the development set. In the case of the Gaussian, we also set the mean to zero.

Tables 5.3 and 5.4 give F scores for NER and accuracies for POS tagging, respectively, for the different priors. The unregularised STANDARD model on each task is included for comparison. The general points are as follows:

1. In order to obtain the results shown in the tables, extensive search of the hyperparameter space is required for each prior family. Typically this required testing 15-20 distinct hyperparameter values, and significant performance improvement can be gained from this careful search.

2. For each prior the regularised CRF significantly outperforms the unregularised

[2]Note that we sometimes use *prior* to mean *prior family*. The meaning should be clear from the context.

STANDARD model. This is no surprise, and is the reason for using a prior at the outset. It does, however, confirm that each prior family is performing as expected.

As with the feature cutoff results earlier, the more specific patterns in the results vary with task. On NER, reading down from Gaussian to Laplacian to Hyperbolic, we observe decreasing F scores on both the development and test sets. However, the only two scores that represent a *significant* difference in performance at the $p < 0.05$ level are those of the Gaussian and Hyperbolic priors on the development set. All other relevant score pairs do not differ significantly. It seems, then, that for NER all the priors perform roughly equally well. For POS tagging the trends are a slightly different. Again the scores for the Gaussian and Laplacian priors do not represent significant differences in performance. The Hyperbolic prior, however, does perform significantly worse than the other two on both the development and test sets. One reason for this may relate not to the regularising effect of the Hyperbolic prior itself, but to the practical difficulty of finding an optimal hyperparameter value for this prior. Glancing back to Equations 5.7 and 5.8, we see that the Hyperbolic prior hyperparameter, β_k, influences the regularisation of the model parameters through a function of exponentials. This function has a much more complicated form than the corresponding functions for the Gaussian and Laplacian priors, and may make manual search for the optimal hyperparameter setting more difficult than with the other priors.

From these results we conclude that the Gaussian prior, although by far the most commonly used, is *not* a natural choice of prior for CRFs on sequencing labelling tasks, and possibly in general. The Laplacian and Gaussian priors obtain very similar performance levels on both NER and POS tagging. The Hyperbolic prior does appear to underperform the other two in some cases, but it not clear whether this relates to some intrinsic property of the prior itself, or just a greater difficulty in its effective application. These conclusions are somewhat contrary to the findings of similar work conducted by Peng and McCallum (2004). They compare the same prior families on an information extraction, using CRFs to extract specific data from research papers. They find that the Gaussian prior performs significantly better than alternative priors on the task and consequently conclude the the Gaussian is the prior of choice. In addition, they report performance figures for the Hyperbolic and Laplacian priors that were lower even than those of the unregularised STANDARD model. There are several possible reasons for these differences. The first is that for the Hyperbolic prior, Peng and McCallum (2004) did not use an adjustable hyperparameter. They instead applied a

discount to each empirical expected feature count which was dependent only on the current value of the respective model parameter and corresponds in our case to using a fixed value of 1 for the β hyperparameter. Our results for this value of the hyperparameter are similarly poor. The second reason is that for the Laplacian prior, they again used a fixed value for the hyperparameter, calculated via an absolute discounting method used language modelling (Chen and Rosenfeld, 2000). Having achieved poor results with this value they experimented with other values but obtained even worse performance.

5.4 Feature Cutoff with a Prior

In section 5.1 we looked at the **feature cutoff** approach to regularisation and found that, at least for the tasks we considered, the approach does not lead to significant benefit for CRFs. We conjectured that by removing low frequency features from the model, we are removing useful features as well as the less useful features that may be a source of overfitting. However, it is possible that other features of frequencies higher than those we considered in section 5.1 also have an overfitting effect to some degree. If this is the case, we may obtain some benefit from applying a combination of feature cutoff and the prior approach we looked at in the last section. We look at this combination here in this section.

In the last section we concluded that the Gaussian and Laplacian priors perform roughly equally well on the two tasks we are considering, with the Hyperbolic prior underperforming the other two in some circumstances. Therefore, rather than presenting results for all priors with a feature cutoff, we take the Gaussian as a representative prior.

Tables 5.5 and 5.6 give F scores and accuracies on NER and POS tagging, respectively, for feature cutoff models with the addition of a Gaussian prior. In every case we observe improved scores over the corresponding models without the prior. As for how the new models compare to the unregularised STANDARD, the trends once again differ with task. For NER none of the models differ in performance significantly from the unregularised STANDARD, except the model with features of frequency three or less removed, and even then only on the development set. However, *all* models significantly underperform the STANDARD model with a Gaussian prior (from the last section), so there seems to be no advantage in combining a feature cutoff with a prior over a prior alone. For POS tagging, the results are more positive. Here *every* model

Frequencies Removed	Development	Test
STANDARD	88.21	81.60
1	88.91	82.56
1 and 2	88.10	82.41
1, 2, and 3	87.55	82.00

Table 5.5: F scores for feature cutoffs with a Gaussian prior on NER.

Frequencies Removed	Development	Test
STANDARD	96.76	97.32
1	97.07	96.66
1 and 2	97.01	96.62
1, 2 and 3	96.92	96.57

Table 5.6: Accuracies for feature cutoffs with a Gaussian prior on POS tagging.

now significantly outperforms the unregularised STANDARD, so the combination is having a useful regularising effect. In addition, results for the first two cutoff categories also do not significantly underperform the STANDARD model regularised with a Gaussian prior.

From these results we conclude that features other than just those with low frequency contribute to model overfitting, and by regularising these features with a prior we can gain some benefit. However, there is no advantage in using a feature cutoff in addition to a straight prior, as we do not obtain significance improvement when doing so.

5.5 Feature-Dependent Regularisation

In section 5.3 we considered what we call **feature-independent regularisation**, where the model parameters associated with different features are regularised to the same degree. This was achieved, when using the priors of section 5.2, by setting the adjustable hyperparameter of the prior to the same value for each model parameter. However, in general it may not be the case that we want to regularise each parameter equally. As we noted in the introduction, we may expect that a parameter associated with a fea-

Threshold	Development	Test
STANDARD	88.21	81.60
Gaussian	89.86	83.97
1	89.00	82.62
2	88.99	83.03
5	89.14	82.81
7	89.29	83.12

Table 5.7: F scores for frequency cutoff on NER.

ture that has been seen many times in the training data may not require regularising to the same degree as a parameter associated with a very infrequently occurring feature. Therefore, a more sophisticated approach to regularisation with a prior would be to regularise different features to different degrees. For this, we may envisage a spectrum of regularisation possibilities. At one end of the spectrum we have the case that we have already considered, where all parameters are regularised to the same degree. We have already denoted this as **feature-independent regularisation**. At the other end of the spectrum we have the opposite case, where regularisation is at the granularity of each individual parameter. We call this **individual feature regularisation**. Somewhere in the middle of the spectrum we have intermediate cases, where parameters are grouped into classes and regularised to the same degree within a class, but to different degrees across classes. We call this **clustered feature regularisation**. In this section we consider this spectrum, investigating the two additional forms of regularisation in the subsections below.

5.5.1 Clustered Feature Regularisation

We consider two methods of clustering features for regularisation with the clusters. We call these approaches **frequency cutoff** and **frequency bins**.

5.5.1.1 Frequency Cutoff

When we considered feature cutoffs in section 5.1, we made the assumption that low frequency features are liable to overfit to such a degree that it may be best to remove all such features from the model. From the results in that section, we concluded that although some low frequency features may exhibit a tendency to overfit to a greater

Threshold	Development	Test
STANDARD	96.76	96.32
Gaussian	97.07	96.69
1	96.97	96.49
2	96.98	96.49
5	96.98	96.56
7	97.01	96.58

Table 5.8: Accuracies for frequency cutoff on POS tagging.

degree than features that occur more often, they do nevertheless contribute useful modelling capability, and we should try to avoid removing them from the model. We could instead retain these features, but regularise them to a greater degree than the other, more frequent features. This method, which we call **frequency cutoff**, specifies a threshold on the frequencies of features in the training data. Parameters associated with features that have frequencies less than or equal to the threshold are all regularised to the same degree. All other parameters (associated with more frequent features) are left unregularised. We use the Gaussian prior as a representative prior for our regularisation, and set the adjustable hyperparameter (the Gaussian variance) for the low frequency features using the development set.

Tables 5.7 and 5.8 give F scores on NER and accuracies on POS tagging, respectively, for models with feature cutoff of varying threshold. The STANDARD model regularised with a Gaussian prior across all the parameters is also included for comparison. The main points from the results are as follows:

1. Generally, as the threshold is increased results improve.

2. On NER, with the exception of the model with a frequency cutoff of one, all models significantly outperform the unregularised STANDARD model at the $p < 0.05$ level. However, all models also significantly underperform the STANDARD model regularised with a Gaussian prior across all the parameters.

3. On POS tagging, *all* models significantly outperform the unregularised the STANDARD model at the $p < 0.05$ level. In addition, some models do not significantly underperform STANDARD model regularised with a Gaussian prior across all the parameters.

These results suggest that the method of regularising low frequency features with a prior is more effective than applying a simple feature cutoff, which by comparison is much more coarse. In addition, it seems that gains can be made by regularising more features in the model than just those with low frequency. This supports our conclusion from section 5.4, that features with frequencies higher than those considered here and in section 5.1 contribute to model overfitting and benefit from some level of regularisation.

5.5.1.2 Frequency Bins

With the **frequency cutoff** approach in the last section we only regularised parameters of features with training data frequencies falling below some threshold. From the results we concluded that features with higher frequencies, although likely to cause overfitting to a lesser degree than rarer features, may nevertheless overfit to some extent. A generalisation of the approach in the previous section, therefore, is to create a number of different feature clusters or **bins**, based on training data frequency, and regularise the associated parameters in each bin to a different degree. With a large number of bins the search space clearly becomes very large. In order to investigate this idea in principle, we constrain the number of bins to be fairly small.

Tables 5.9 and 5.10 give F scores on NER and accuracies on POS tagging, respectively, for representative frequency bins. In this case features of frequencies ten or less are divided into two bins: frequencies $1-3$ and frequencies $4-10$. We allocate a separate adjustable hyperparameter to each bin, and optimise manually using the development set. From the tables we observe similar, but improved, results over the models in the previous section. In all cases, for both tasks, the frequency bin models significantly outperform the unregularised STANDARD model. In addition, a greater number of models than in the previous section do not significantly underperform the STANDARD model regularised with a single Gaussian prior across all parameters. This suggests that applying a more specific level of regularisation to "similar" features is advantageous, where we are basing similarity here on training data frequency. The extreme case would be to regularise each feature to its own specific degree. Clearly this would make the hyperparameter search space intractably large. However, it might be possible instead to fix the *ratio* of the level of regularisation applied to different parameters and adjust the absolute level of regularisation via this ratio and a smaller number of hyperparameters, just one in the extreme case. We follow these ideas in the section 5.5.2. However, before that we consider another way of clustering the features

σ_{1-3}	σ_{4-10}	Development	Test
∞	∞	88.21	81.60
197	210	89.04	83.15
210	217	88.98	83.34
191.2	228.8	89.06	83.19

Table 5.9: F scores for frequency bins on NER.

σ_{1-3}	σ_{4-10}	Development	Test
∞	∞	96.76	96.32
1.781	2.3	97.05	96.58
2.3	3.11	97.03	96.60
1.915	2.685	97.03	96.58

Table 5.10: Accuracies for frequency bins on POS tagging.

into subsets for regularisation. Instead of using training data frequency, we cluster based on the kind of dependency different features encode.

5.5.1.3 Dependency Types

With both the **frequency cutoff** and **frequency bins** approaches above we cluster the features into groups for regularisation based directly on the frequency of the features in the training data. In this section we modify this idea slightly by clustering the features based on the type of dependency they encode. At a high level the features we use in the STANDARD CRF model (on both tasks) may be divided into two types: those that model *label-label* dependencies and those that model *label-observation* dependencies. Features in the first set model the dependency of a word's label on the previous word's label, while features in the second set model the dependency of a word's label on properties of the word itself and possibly other information contained in the observations, like POS tags for words in a local neighbourhood of the word. These two sets of features can be thought of as roughly corresponding to **transitions** and **emissions** in hidden Markov models, and we will therefore refer to the two sets as "transitions" and "emissions" in this section. Table 5.11 illustrates the two sets of features for NER. We see that the mean count for "emissions" is less than that for "transitions". A sim-

Feature Set	Number of Features	Mean Feature Count
"transition"	29,726	12.12
"emissions"	159,613	6.71

Table 5.11: Properties of "transitions" and "emissions" feature sets on NER.

σ_t	σ_e	Development	Test
∞	∞	88.21	81.60
101.4	44	90.18	83.69
44	18	90.09	83.83
75	25	90.05	83.57

Table 5.12: F scores for dependency types on NER.

ilar story is true for POS tagging. Because the two sets of features encode different dependencies with correspondingly different mean counts, it may be appropriate to regularise them separately. We do this as before by allocating an independent Gaussian variance hyperparameter to each feature set and adjusting each hyperparameter separately. In order to make the hyperparameter search manageable, we use the global optimal Gaussian variance (across all parameters) as a starting point and adjust the separate "transition" and "emission" variances around this.

Tables 5.12 and 5.13 give F scores on NER and accuracies on POS tagging, respectively, for representative dependency type models. In the tables σ_t is the square root of the Gaussian variance hyperparameter for the "transitions" and σ_e is the square root of the Gaussian variance for the "emissions". The main points from the results are as follows:

1. All models, on both tasks, significantly outperform the unregularised STANDARD model.

2. No model, on either task, significantly underperforms the STANDARD model regularised with a single Gaussian prior across all parameters.

These results support our idea that clustering features based on the role they play, or dependencies they encode, and regularising different clusters separately, can be beneficial. Of course, this approach to clustering is effectively orthogonal to the one we

σ_t	σ_e	Development	Test
∞	∞	96.76	96.32
5.1	3.2	97.10	96.73
3.2	2.4	97.11	96.74
3.7	2.7	97.11	96.73

Table 5.13: Accuracies for dependency types on POS tagging.

σ	Development	Test
STANDARD	88.21	81.60
620	89.20	82.97

Table 5.14: F scores for frequency-based individual feature regularisation on NER.

were taking in previous sections, which was based on the frequency of the features in the training data. Therefore, in principle, the two could be combined, although once again we would have the problem of a potentially very large hyperparameter search space. The advantage of the clustering method presented in this section is that there are only two clusters, yet we obtain results that are in some cases better (although not significantly at $p < 0.05$) than those of the STANDARD model regularised with a single Gaussian prior across all the parameters.

5.5.2 Individual Feature Regularisation

Recalling the hypothetical spectrum we described earlier, at one end of the spectrum we have the standard case for regularisation (described in section 5.3) where all parameters are regularised to an equal degree. In this section we move to the other end of the spectrum, and look at regularisation at the level of granularity of the individual parameter. We consider two ways in which we might do this. The first is based solely on feature frequency (like the **frequency cutoff** and **frequency bins** approaches earlier). The other involves feature frequency, but is a little more sophisticated and takes context information into account.

σ	Development	Test
STANDARD	96.76	96.32
0.87	97.03	96.54

Table 5.15: Accuracies for frequency-based individual feature regularisation on POS tagging.

5.5.2.1 Frequency-Based

In principle we could stretch the **frequency bins** approach above to the extreme case where each bin has a width of just one, so each parameter is regularised to a different degree and we set each of the hyperparameters independently using the development set. Clearly, this would make the hyperparameter search space infeasibly large. However, as we alluded to earlier, we could instead constrain the *ratio* by which each model parameter is regularised with respect to all others, and allow the absolute degree of regularisation to vary with a *single* independent variable against this ratio. We can then set this single independent variable using the development set. The task, then, would be to define the ratio by which different model parameters are regularised with respect to each other. We base this ratio on training data frequency of the associated features, as in earlier sections. To see how we may do this, consider again the following set of equations (explained in section 5.2.1) that hold at the optimal point when applying a Gaussian prior:

$$E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] = \frac{\lambda_k - \mu_k}{\sigma_k^2} \tag{5.9}$$

Assuming a Gaussian mean μ_k of zero, we have a right-hand side of λ_k/σ_k^2, with the variance σ_k^2 defining the level of regularisation for model parameter λ_k. Now, if we set $\sigma_k^2 = c_k\sigma^2$ for a single independent variable σ, we constrain the relative amount by which a model parameter is regularised in a way that depends on the count, c_k, of the associated feature in the training data. So the higher the training data feature count, the lower the degree of regularisation of the model parameter. The absolute level of regularisation for each model parameter is then defined by setting the value of σ, which can be done using the development data.

Tables 5.14 and 5.15 give F scores on NER and accuracies on POS tagging, respectively, for frequency-based individual feature regularisation (FB-IFR). The σ val-

Regularisation	Development	Test
STANDARD (None)	88.21	81.60
FCB-IFR	88.84	82.84

Table 5.16: F scores for feature and context-based individual feature regularisation on NER.

Regularisation	Development	Test
STANDARD (None)	96.76	96.32
FCB-IFR	96.87	96.39

Table 5.17: Accuracies for feature and context-based individual feature regularisation on POS tagging.

ues are given in each case. We can see from the tables that there is a large difference in magnitude between the σ value for NER and that for POS tagging. However, this difference is not particularly important – we would not expect the two σ values to be similar necessarily because they refer to different tasks, with datasets and feature sets of different sizes.

The results in the tables show that in all cases the FB-IFR models significantly outperform the unregularised STANDARD model at the $p < 0.05$ level. However, except for one case, the models also significantly underperform the STANDARD model regularised with a single Gaussian prior across all the parameters. What we are doing here is very similar to applying a Gaussian prior with a single variance across all parameters, but in such a way that the effect on each parameter is skewed using the feature count. Given that the results we obtain are worse than with a single Gaussian prior, it is possible that the feature count is not the most appropriate factor to use in determining the regularisation ratio. In the next section we consider a slightly more sophisticated method for determining the ratio.

5.5.2.2 Frequency and Context-Based

In many of the previous sections we have used the frequency of a feature in the training data to determine the level of regularisation to be applied to an associated model parameter. This was motivated by the idea that infrequent features are less representative

of the underlying distribution than more commonly occurring ones, and so are more likely to overfit to the idiosyncratic properties of the training data. Although this idea is reasonable at a high level, it is slightly simplistic and a little crude. In this section we take a more sophisticated approach from a Bayesian viewpoint.

Recalling Equation 5.3, we see that the level of regularisation applied to a model parameter takes the form of a discount to the expected count of the associated feature on the training data. It is natural, therefore, that the size of this discount, controlled through the hyperparameter σ_k, is related to our confidence in the *reliability* of the empirical expected count. We must therefore formulate a measure of this confidence. We follow the approach of Kazama and Tsujii (2003), who developed a method to do this for a standard maximum entropy model. We extend the method here to CRFs.

The empirical expected count $E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k]$ in Equation 5.3 of a feature f_k is given by:

$$\begin{aligned}\sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \sum_t f_k(s_{t-1},s_t,\mathbf{o},t) &= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \tilde{p}(\mathbf{s}|\mathbf{o}) \sum_t f_k(s_{t-1},s_t,\mathbf{o},t) \\ &= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{t,s',s''} \tilde{p}(s_{t-1}=s',s_t=s''|\mathbf{o})\, f_k(s',s'',\mathbf{o},t)\end{aligned} \tag{5.10}$$

Now, our CRF features (see Chapter 3) generally have the following form:

$$f_k(s_{t-1},s_t,\mathbf{o},t) = \begin{cases} 1 & \text{if } s_{t-1}=s_1,\ s_t=s_2 \text{ and } h_k(\mathbf{o},t)=1 \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

where s_1 and s_2 are the labels associated with feature f_k and $h_k(\mathbf{o},t)$ is a binary-valued predicate defined on observation sequence $\mathbf{o}$ at position t. With this feature definition, and contracting notation for the empirical probability to save space, $E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k]$ becomes:

$$\begin{aligned}\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{t,s',s''} \tilde{p}(s',s''|\mathbf{o})\delta(s',s_1)\delta(s'',s_2)h_k(\mathbf{o},t) &= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_t \tilde{p}(s_1,s_2|\mathbf{o})h_k(\mathbf{o},t) \\ &= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{t:h_k(\mathbf{o},t)=1} \tilde{p}(s_1,s_2|\mathbf{o})\end{aligned}$$

Contributions to the inner sum are only made at positions t in sequence $\mathbf{o}$ where the $h_k(\mathbf{o},t)=1$. Suppose that we make the assumption that at these positions $\tilde{p}(s',s''|\mathbf{o}) \approx \tilde{p}(s',s''|h_k(\mathbf{o},t)=1)$. Then:

$$E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] = \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{t:h_k(\mathbf{o},t)=1} \tilde{p}(s_1,s_2|h_k(\mathbf{o},t)=1) \tag{5.12}$$

Now, if we assume that we can get a reasonable estimate of $\tilde{p}(\mathbf{o})$ from the training data then the only source of uncertainty in the expression for $E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k]$ is the term $\tilde{p}(s_{t-1} = s_1, s_t = s_2 | h_k(\mathbf{o},t) = 1)$. Since this term is independent of sequence $\mathbf{o}$ and position t, we can model it as the parameter θ of a Bernoulli random variable that takes the value 1 when feature f_k is active and 0 when the feature is not active but $h_k(\mathbf{o},t) = 1$. Suppose there are a and b instances of these two events, respectively. We endow the Bernoulli parameter with a uniform prior Beta distribution Be(1,1) and, having observed the training data, we calculate the variance of the posterior distribution, Be($1+a$, $1+b$). The variance is given by:

$$\text{var}[\theta] = V = \frac{(1+a)(1+b)}{(a+b+2)^2(a+b+3)} \tag{5.13}$$

The variance of $E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k]$ therefore given by:

$$\text{var}\left[E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k]\right] = V\left[\sum_{\mathbf{o}} \sum_{t:h_k(\mathbf{o},t)=1} \tilde{p}(\mathbf{o})^2\right] \tag{5.14}$$

We may use this variance as a measure of the confidence we have in $E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k]$ as an estimate of the true expected count of feature f_k. We therefore adjust the Gaussian variance σ_k^2 for each model parameter λ_k according to this confidence measure for the associated feature. Each such confidence measure is not variable at runtime time, so can be calculated once off-line.

Tables 5.16 and 5.17 give results for feature and context-based individual feature regularisation (FCB-IFR). From the tables, we see that the scores are worse than those obtained with the FB-IFR models of the previous section. Only two of the four FCB-IFR models significantly outperform the unregularised STANDARD model at the $p < 0.05$ level, and *all* significantly underperform the STANDARD model regularised with a single Gaussian prior across all parameters.

In this section and the last we have looked at ways in which we may regularise a model at the level of individual features while at the same time avoiding the need to fit a large number of independent hyperparameters. Although other approaches to achieving this goal clearly exist, the results of the last two sections suggest the task is not easy. Our findings in earlier sections in this chapter suggested that improvements could be made with more careful feature-specific regularisation, at the level of feature clusters or individual features. However, combined with the results in the last two sections it seems we cannot easily achieve this objective without the inevitable trade-off

Mean	Development	Test
0	89.86	83.97
0.1	90.16	84.03
0.5	90.44	84.68
0.6	**90.47**	**84.43**
1	90.35	84.51
5	87.43	81.36
10	87.16	81.13

Table 5.18: F scores for positive Gaussian prior means on NER.

Mean	Development	Test
0	**89.86**	**83.97**
−0.1	89.64	83.72
−0.5	89.11	83.20
−1	88.65	82.83
−5	88.26	82.51
−10	88.07	82.38

Table 5.19: F scores for negative Gaussian prior means on NER.

with a hyperparameter search space of high dimension. The compromise, therefore, may be an approach which lies somewhere in the middle of the hypothetical spectrum we described earlier. An example of such a compromise would be the **dependency types** method of section 5.5.1.3, where we achieved better results than with a single Gaussian prior over all the parameters, but using only two independent hyperparameters.

5.6 Varying the Gaussian Mean

In the last few sections we have considered how we may apply different levels of regularisation to the features in a model. In each case we were using a Gaussian prior over clusters of features and controlling the level of regularisation applied to each cluster by adjusting the Gaussian variance associated with that cluster. In this section

Mean	Development	Test
0	97.07	96.69
−5	95.80	95.46
−1	96.86	96.48
−0.1	97.02	96.68
0.1	97.14	96.77
0.67	**97.37**	**96.97**
1	97.31	96.88
5	96.14	95.68

Table 5.20: Accuracies for non-zero Gaussian prior means on POS tagging.

we move on to consider another aspect of the Gaussian prior: the effect of varying the mean of the distribution. When using a Gaussian prior it is usual to fix the mean at zero because there is usually no prior information to suggest that we should penalise large positive values of model parameters any more or less than large magnitude negative values. It also simplifies the hyperparameter search, requiring the need to optimise only the variance hyperparameter (or hyperparameters) rather than the mean and variance jointly. However, it is unlikely that optimal performance is always achieved for a mean value of zero. For example, particularly where only *supported* features[3] are instantiated, the features are likely to represent conjunctions of predicates defined on the observations and labels that represent events that are likely to occur. This will usually result in more features having associated parameters with positive values than negative. This would typically result in parameter values with a positive mean.

To investigate this we allow the Gaussian mean to vary away from 0. In order to simplify the hyperparameter search and avoid having to search the joint mean and variance space, we use the variance at the optimal value we found earlier in section 5.3 (with a mean of zero). We then fix the variance at this value and allow the mean to vary away from zero.

Tables 5.18 and 5.19 give F scores on NER for models trained in this way, with non-zero means. Table 5.18 shows models with positive means, while Table 5.19 shows negative means. From the tables we observe that:

1. Models with negative values for the Gaussian mean perform worse than the zero-

[3] *Supported* features are those that are observed in the training data (as explained in Chapter 2).

mean model. This is not surprising because, as we noted above, we expect the optimal mean value to be positive.

2. Models with a range of small positive values for the Gaussian mean outperform the zero-mean model.

3. The model with the optimal value for the Gaussian mean on NER has a mean of 0.6. This significantly outperforms the zero-mean model at the $p < 0.05$ level.

The pattern we see in Tables 5.18 and 5.19 for NER is also observed for POS tagging. Table 5.20 gives illustrative results on this task, with both positive and negative values of the Gaussian mean. The optimal value for the mean is once again a small positive value, this time around 0.67. These results confirm our conjecture about the non-zero values for the Gaussian mean, and suggest that considerable benefit may be gained from a well structured search of the joint mean and variance hyperparameter space when using a Gaussian prior for regularisation. Once again, however, there is of course a trade-off here between finding better hyperparameter values and suffering increased search complexity.

5.7 The Inequality CRF

In this section we consider an alternative approach to CRF regularisation that addresses the problem from a slightly different starting point. We derive a variant of the vanilla CRF that has a regularising effect implicit in its form. We call this new model the **inequality CRF**. Our model is an extension of a similar idea applied to maximum entropy models by Kazama and Tsujii (2003).

In Chapter 2 we saw that when training the parameters of a CRF to maximise the conditional log-likelihood of the training data, at the optimal point a set of **equality constraints** on feature expectations are satisfied. These have the form:

$$E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] = 0, \ \forall k \tag{5.15}$$

Here the index k runs over all features $1 \ldots K$ in the model, so we have one such constraint for each feature. These constraints enforce a strong condition on the model: that the expected count of each feature under the model is equal to the count of that feature on the training data (the feature's empirical expected count). In practice we may regard

this condition as too strong in cases where we have a relatively small training set. For example, for a sparse feature we may not believe that the empirical expected count (calculated from the small training set) is a reliable estimate of that feature's regularity of occurrence in general. Consequently, we may not want to enforce the constraint (from the set described in Equation 5.15) for that particular feature. Instead, we may want to *relax* the constraint and allow the feature's empirical expected count to act only as a rough guide to what the expected count under the model should be.

When considering different families of priors earlier in this chapter, we saw that for each feature a prior enforces the condition that the model's expected count of the feature be *discounted* with respect to the feature's empirical expected count. This is a kind of relaxing of the constraints in Equation 5.15. In this section, however, we consider a different form of relaxation of these constraints. In particular, for each feature we enforce the constraint that the model's expected count for the feature lie within a specified window *around* the feature's empirical expected count (but not that the two are necessarily equal). We therefore enforce a set of conditions of the form:

$$A_k \geq E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] \geq -B_k, \;\; A_k, B_k > 0 \tag{5.16}$$

In 5.16 the A_k and B_k control the size of the window around the empirical expected count for feature f_k. So, if we have little confidence that the empirical expected count is accurate for f_k, we relax that constraint from the set in Equation 5.16 more by allowing A_k and B_k to be larger. Conversely, if we have greater confidence in the accuracy of the empirical expected count for f_k, we relax the constraint less by making A_k and B_k smaller. In the extreme case where we have absolute confidence in the empirical expected count of all features, the set of constraints in Equation 5.16 collapses to the original set of constraints in Equation 5.15. This is saying that the training data is a perfect predictor for any future data, and we do not need any smoothing.

Using the set of constraints described in Equation 5.16 as a starting point, we may define a modified CRF by solving an optimisation problem. We call this modified CRF the **inequality CRF**, and describe it in the next section.

5.7.1 Modified Model

In Chapter 2 we defined a linear chain CRF as a model with the following distribution:

$$p(\mathbf{s} \mid \mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp\left(\sum_{t=1}^{T+1} \sum_{k=1}^{K} \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t) \right) \tag{5.17}$$

We show in Appendix A that this general form of CRF distribution is the solution to a sequential maximum entropy problem that includes feature expectation constraints of a form similar to those in Equation 5.15. This optimisation problem can be stated as follows:

$$\begin{aligned} &\max_{p(\mathbf{s}|\mathbf{o})} \mathrm{H}[p(\mathbf{s} \mid \mathbf{o})] = -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o}) \\ &\text{s.t.} \quad E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] = 0, \;\; k = 1 \ldots K \end{aligned} \tag{5.18}$$

Following our argumentation in the previous section regarding relaxation of the feature expectation constraints, we now define a modified optimisation problem that includes the constraints given in Equation 5.16. Our new problem is therefore:

$$\begin{aligned} &\max_{p(\mathbf{s}|\mathbf{o})} \mathrm{H}[p(\mathbf{s} \mid \mathbf{o})] = -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o}) \\ &\text{s.t.} \quad E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] - A_k \leq 0, \;\; k = 1 \ldots K \\ &\qquad E_{p(\mathbf{s}|\mathbf{o})}[f_k] - E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - B_k \leq 0, \;\; k = 1 \ldots K \end{aligned} \tag{5.19}$$

In Appendix B we show that the general solution to this problem has the form:

$$p(\mathbf{s}|\mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp\left(\sum_{t=1}^{T+1} \sum_{k=1}^{K} (\alpha_k - \beta_k) \, f_k(s_{t-1}, s_t, \mathbf{o}, t) \right) \tag{5.20}$$

where the parameters α_k and β_k are non-negative Lagrange multipliers that satisfy the Karush-Kuhn-Tucker (KKT) conditions (Nocedal and Wright, 1999):

$$\begin{aligned} \alpha_k \left(E_{\tilde{p}}[f_k] - E_p[f_k] - A_k \right) &= 0, \;\; k = 1 \ldots K \\ \beta_k \left(E_p[f_k] - E_{\tilde{p}}[f_k] - B_k \right) &= 0, \;\; k = 1 \ldots K \end{aligned} \tag{5.21}$$

We call this model the **inequality CRF**. Note that the model has twice the number of parameters as a standard CRF defined using the same feature set. Each model parameter λ_k from the standard CRF is effectively "split" into two independent non-negative variables, α_k and β_k.

5.7.2 Training

To train the new model we use a modified conditional log-likelihood objective. We form the dual objective function corresponding to the solution in Equation 5.20 of the optimisation problem in 5.19, and obtain (see Appendix B for details):

$$\Lambda(\alpha,\beta) = LL(\alpha,\beta) - \sum_k \alpha_k A_k - \sum_k \beta_k B_k \tag{5.22}$$

$$\alpha_k \geq 0, \;\; \beta_k \geq 0, \;\; k = 1 \ldots K \tag{5.23}$$

We then maximise this adjusted conditional log-likelihood, with the parameters constrained to satisfy the inequalities in 5.23. During the optimisation we require the gradient of the new objective. We know the form for the gradient of the log-likelihood term in Equation 5.22 from a standard CRF, and the derivatives of the latter two terms are trivial to evaluate. Therefore the gradient of the objective function is given by:

$$\frac{\partial \Lambda(\alpha,\beta)}{\partial \alpha_k} = E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] - A_k \tag{5.24}$$

$$\frac{\partial \Lambda(\alpha,\beta)}{\partial \beta_k} = E_{p(\mathbf{s}|\mathbf{o})}[f_k] - E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - B_k \tag{5.25}$$

5.7.3 Quadratic Penalty Extension

The derivation of the inequality CRF above results in a model that is similar to a standard CRF with a prior (we explain this a little later, in section 5.7.6). However, we may also include a quadratic penalty to the "new" parameters α_k and β_k which is similar to explicitly defining a Gaussian prior over these parameters. Following the terminology of Kazama and Tsujii (2003), who applied the same idea to a maximum entropy model, we refer to this modified model as the inequality CRF with a **quadratic penalty extension (QPE)**. In the extension the inequality constraints 5.16 are modified to include slack variables γ_k and δ_k, and the objective is altered to include penalty terms quadratic in these variables. The new optimisation problem therefore becomes:

$$\begin{aligned} &\max_{p,\delta,\gamma} H(p) - C_1 \sum_k \delta_k^2 - C_2 \sum_k \gamma_k^2 \\ &\text{s.t.} \quad E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] - A_k \leq \delta_k \\ &\qquad E_{p(\mathbf{s}|\mathbf{o})}[f_k] - E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - B_k \leq \gamma_k \end{aligned} \tag{5.26}$$

where C_1 and C_2 are constants. The general solution to this problem has the same form as the basic inequality CRF. However, the dual objective function is different, and becomes:

$$\Lambda'(\alpha,\beta) = \Lambda(\alpha,\beta) - \sum_k \left(\frac{\alpha_k^2}{4C_1} + \frac{\beta_k^2}{4C_2} \right) \tag{5.27}$$

The gradient of the objective correspondingly becomes:

$$\frac{\partial \Lambda'(\alpha,\beta)}{\partial \alpha_k} = E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] - \left(A_k + \frac{\alpha_k}{2C_1} \right) \tag{5.28}$$

$$\frac{\partial \Lambda'(\alpha,\beta)}{\partial \beta_k} = E_{p(\mathbf{s}|\mathbf{o})}[f_k] - E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - \left(B_k + \frac{\beta_k}{2C_2} \right) \tag{5.29}$$

We can see that the additional terms entering into the objective and its gradient are similar to the corresponding terms when using a Gaussian prior with a standard CRF. Therefore the quadratic penalty extension simulates the addition of a Gaussian prior to the inequality CRF. In effect we have two levels of regularisation, the implicit regularisation provided by the inequality CRF itself, and explicit regularisation provided by the Gaussian-style prior.

5.7.4 Bounded Gradient-Based Optimisation

From Equation 5.20 we see that, in contrast to estimation for a standard CRF in Equation 5.17, the optimisation problem now involves bounds on the parameters, each parameter being non-negative. We must therefore use an optimisation routine that provides this facility. In our experiments, we use a bounded variant of limited memory variable metric (LMVM), called **bounded limited memory variable metric (BLMVM)** (Benson et al., 2002). We use an implementation of the algorithm that is also contained in the Toolkit for Advanced Optimisation (TAO) (Benson and More, 2001) (see Chapter 3).

5.7.5 Model Sparsity

The inequality CRF has the tendency to produce sparse solutions. This is somewhat similar to the situation with the Laplacian prior we discussed earlier in section 5.2.2. To see why this sparseness occurs, note that from the KKT conditions in 5.21, because $A_k,\ B_k > 0$, if α_k is non-zero then β_k is zero, and vice versa. Hence either α_k or

β_k, or both, are zero. When both are zero, the corresponding parameter is zero and the solution becomes sparse if this occurs for many parameters. Also from the KKT conditions, it can be seen that where either α_k or β_k is non-zero, the corresponding equality constraint is most relaxed. The inequality constraints define a feasible region within the entire parameter space. Cases where both α_k and β_k are zero correspond to points within the region, while cases where either α_k or β_k are non-zero correspond to points on the boundary of the region. As A_k and/or B_k are increased, the equality constraints are relaxed further and the feasible region is enlarged. Hence increasing the value of A_k and/or B_k tends to make the solution to the optimisation problem in 5.19 more sparse.

5.7.6 Relationship to the Exponential Prior

The inequality CRF shares a strong connection with the **Exponential** prior, which is a one-sided version of the Laplacian prior we saw earlier in section 5.2.2. The Laplacian prior and the inequality CRF are therefore also related. We discuss the relationship between the inequality CRF and Exponential prior in this section.

For positive parameter α, an exponential distribution has a probability density function:

$$p(x) = \left\{ \begin{array}{ll} \alpha e^{-\alpha x} & \text{if } x > 0 \\ 0 & \text{otherwise} \end{array} \right\} \tag{5.30}$$

Consequently, when using this distribution as a prior, parameter values are forced to be non-negative. In general, for log-linear models (and CRFs) parameter values may be positive or negative: a positive value indicating evidence for the event represented by the feature and a negative value indicating evidence against the event. As Goodman (2004) shows, one way to reconcile this situation is to create, for every feature f_k, a complementary feature $\tilde{f}_k$ representing the event which is a conditional complement of that represented by the feature. In this case, rather than having one parameter λ_k, for f_k, which could be positive or negative, we have two parameters, λ_k and $\tilde{\lambda}_k$, for f_k and $\tilde{f}_k$ respectively, which are both non-negative. In a sense the negative evidence for f_k would be represented through the positive value $\tilde{\lambda}_k$. Goodman uses this approach to derive GIS update equations for maximum entropy models with an Exponential prior. However, using BLMVM, no ad hoc modification to existing update code is required as the bound constraints are directly handled by the estimation method.

If we assign an Exponential prior distribution to each parameter λ_k and $\tilde{\lambda}_k$, with hyperparameters A_k and B_k respectively, then, ignoring constant terms that do not involve the parameters, the regularised log-likelihood becomes:

$$LL(\lambda - \tilde{\lambda}) - \sum_k \lambda_k A_k - \sum \tilde{\lambda}_k B_k \tag{5.31}$$

$$\lambda_k, \tilde{\lambda}_k \geq 0 \tag{5.32}$$

and the gradient of the objective becomes:

$$\frac{\partial \Lambda(\lambda - \tilde{\lambda})}{\partial \lambda_k} = E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] - A_k \tag{5.33}$$

$$\frac{\partial \Lambda(\lambda - \tilde{\lambda})}{\partial \tilde{\lambda}_k} = E_{p(\mathbf{s}|\mathbf{o})}[f_k] - E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - B_k \tag{5.34}$$

Hence by applying the Exponential prior in this way, we arrive at the same problem as that for the inequality CRF. Given the connection between the Exponential and Laplacian priors, the derivatives for the Exponential prior in Equations 5.33 and 5.34 above are similar in nature to the corresponding derivatives for the Laplacian prior in Equation 5.6.

5.7.7 Window Width

The parameters A_k and B_k in the inequality constraints in 5.16 determine the size of the "window" around the corresponding equality constraint, the amount by which the equality constraint may be violated. We can think of the A_k representing a set of **lower bounds** and the B_k representing a set of **upper bounds**. As we reasoned earlier, the window size should reflect our confidence in the value of the corresponding empirical feature expectation. Therefore, we require a way to calculate this window size. To do this we consider two distinct possibilities, which we call **modes** of the inequality CRF:

- **Mode 1**: The lower and upper window bounds are constrained to be equal for a given feature, and are fixed across features. Hence:

$$A_k = B_k = C, \text{ for all } k \tag{5.35}$$

Mode	A_k	B_k	Development	Test
1	0.0005	0.0005	89.63	83.60
2	0.0005	0.0052	90.08*	84.04

Table 5.21: F scores for different inequality CRF modes on NER.

Mode	A_k	B_k	Development	Test
1	0.197	0.197	97.09	96.73
2	0.197	0.700	97.20*	96.84*

Table 5.22: Accuracies for different inequality CRF modes on POS tagging.

- **Mode 2**: The lower and upper window bounds are allowed to vary independently for a given feature, but are each fixed across features. Hence:

$$A_k = C \text{ and } B_k = D, \text{ for all } k \tag{5.36}$$

In addition to the two modes above, there is also the possibility of using the **quadratic penalty extension**. This is orthogonal to the distinction between the modes, and could in principle be applied to each mode. For our experiments we apply the QPE just to mode 1 as an illustration of the possible improvement from the additional level of regularisation.

5.7.8 Results

Tables 5.21 and 5.22 give F scores for NER and accuracies for POS tagging, respectively, for the two modes of the inequality CRF. Note that we are not using the quadratic penalty extension here. The main points from the results are as follows:

1. For both modes on both tasks the inequality CRF significantly outperforms the unregularised STANDARD model at the $p < 0.05$ level.

2. For both modes on both tasks the inequality CRF does not significantly underperform the STANDARD model regularised with a Gaussian prior. In addition, in three of the four cases (marked with an asterisk in the tables), mode 2 significantly outperforms the STANDARD model regularised with a Gaussian prior.

Mode	Development	Test
Gaussian	89.86	83.97
1	89.63	83.60
1 + QPE	89.97	83.81

Table 5.23: F scores for inequality CRF mode 1 with QPE on NER.

The results for mode 1 are not surprising because, as we saw earlier in the chapter, in its basic form the inequality CRF shares a strong connection to the Laplacian prior of section 5.2. The performance scores for mode 1 are similar to those for the Laplacian (and Gaussian) prior from that section. Mode 2 is the more interesting case. The extra degree of freedom afforded by the independent adjustment of the A_k and B_k bounds allows for significant improvement over the Gaussian and Laplacian priors. With the inequality CRF in mode 2 we have therefore improved on the aims of the **dependency types** approach of section 5.5.1.3: we have been able to significantly improve on the Gaussian prior using two adjustable bounds, or hyperparameters.

For the quadratic penalty extension, we simplify the search for values of the C_1 and C_2 by constraining them to be equal to one independent adjustable parameter. Tables 5.23 and 5.24 give F scores for NER and accuracies for POS tagging, respectively, for the QPE applied to mode 1. From the results we can see that in general scores are higher with the QPE than without it. In addition, on both tasks scores with the QPE are higher than those with the STANDARD model regularised with a Gaussian prior. However, none of the results are significantly better than those of the Gaussian prior at the $p < 0.05$ level. We conclude then that the QPE has positive, but limited usefulness over and above the implicit regularising effect of the inequality CRF in mode 1. Although we do not present results for the QPE applied to inequality CRF in mode 2, we expect that the results would show a similar marginal improvement to those of mode 1.

5.7.8.1 Model Sparsity

In section 5.7.5 we saw how the inequality CRF creates sparse solutions, with many parameters taking zero values. In this section we looking briefly at this effect. Table 5.25 shows how model sparsity varies with window size for the inequality CRF in mode 1 on NER. Similar results can be observed for POS tagging. We see from the table that

Mode	Development	Test
Gaussian	97.07	96.69
1	97.09	96.73
1 + QPE	97.11	96.72

Table 5.24: Accuracies for inequality CRF mode 1 with QPE on POS tagging.

A_k, B_k	Parameter Count	F Score
$1e-06$	431630	87.45
$1e-05$	359783	88.95
$1e-04$	91038	89.43
0.001	29951	89.60
0.01	17206	89.43
0.1	14054	89.06
1	6551	87.04

Table 5.25: Model sparsity for mode 1 on NER.

increasing the window size tends to lead to a more general model, i.e. one that is closer to the uniform distribution. For small values of window size this is advantageous, as we have seen, as there is less overfitting to the training data. However, for larger values performance is adversely affected as the model becomes too sparse.

Figure 5.2 shows how the number of non-zero parameters varies as training progresses for an inequality CRF in mode 1 on NER. From the graph we see that the number of non-zero parameters decreases near-monotonically during training. It may be tempting to conclude from this that a possible efficiency gain for the inequality CRF might be to remove zero-valued parameters from the model early so as to reduce the load on memory. However, we observed that during training some parameters alternated in value between zero and non-zero, making the removal of zero-valued parameters prematurely potentially dangerous.

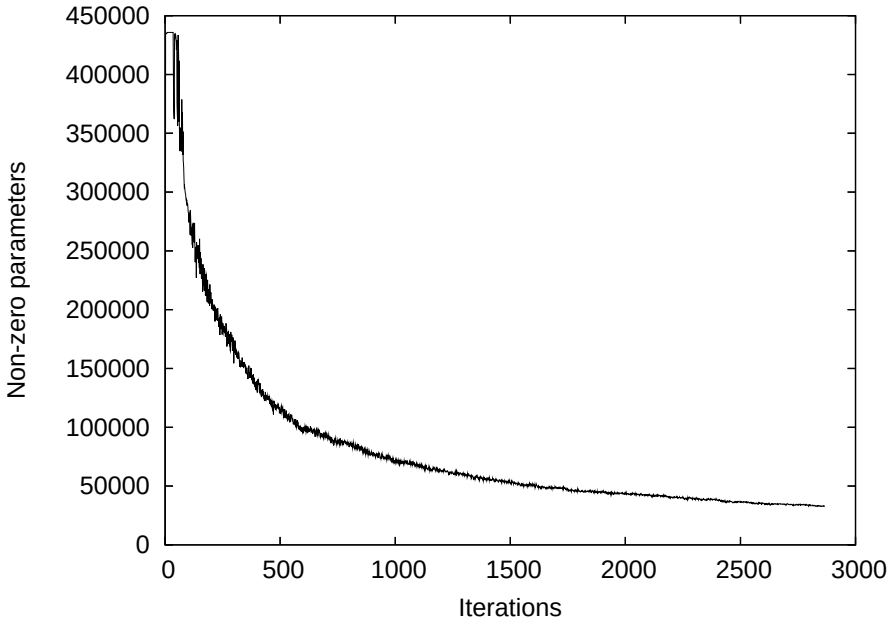

Figure 5.2: Model sparsity during training on NER.

5.8 Summary

In this chapter we have surveyed a number of conventional regularisation techniques for CRFs, and compared their effectiveness. We started by considering very simple approaches such as feature cutoff, and found them to be of little use when used with a CRF. We then looked at the technique of using a prior distribution over the model parameters, and compared three families of priors. Our findings suggest that the Gaussian prior, although the most commonly used with CRFs in practice, is not clearly the most natural choice and other possibilities exist. The Laplacian prior, for example, attains very similar performance levels at no extra cost. We then moved on to consider how to apply regularisation to feature subsets at different levels of granularity. Here our general conclusion is that improvement may be made by applying regularisation at the feature-specific level, whether it be over clusters of features or at the individual feature level. However, this benefit is usually offset by the increased complexity of searching a high-dimensional hyperparameter space. Finally, towards the end of the chapter, we considered an alternative approach to CRF regularisation by formulating a variant of the standard CRF model that has a regularising effect implicit in its form. We showed that this model can lead to significant performance improvement over a standard CRF regularised with a Gaussian prior, at the cost of having to optimise only

one additional hyperparameter.

In the next chapter we move on to consider an alternative framework for CRF regularisation based on the combination of separate CRF models in an ensemble structure. Our aim is to employ *unregularised* CRFs in a more sophisticated way. In particular, we hope to avoid the need to search a high-dimensional hyperparameter space, characteristic of conventional regularisation with a prior.

Chapter 6

Logarithmic Opinion Pools for CRFs

In the last chapter we saw how we may regularise a CRF by applying a prior distribution over the model parameters. We looked at a number of different priors and compared their regularising effects. We also investigated the possibility of regularising different model parameters to differing degrees, looking at a number of ways in which this may be achieved. We ended the chapter by concluding that although some improvements may be made to the conventional approach to CRF regularisation through small modifications and extensions, it would be desirable instead to find a different approach that avoids the need to search a (potentially large) hyperparameter space. In this chapter we introduce such a framework. Our approach is based on a form of CRF *ensemble* called a **logarithmic opinion pool** (LOP). In the course of the chapter we describe properties of the LOP, and compare its effectiveness as a method for overfitting reduction with the priors used in the previous chapter.

Ensembles have been used for a number of years (Perrone and Cooper (1993), Sollich and Kroghtree-s (1995)) and have been shown in many circumstances to reduce generalisation error over that of the constituent models, particularly in cases where the errors of the constituents are uncorrelated (Rosen (1996), Oza and Tumer (2001)). Theoretical justification for this behaviour is provided by the variance reducing properties of the ensemble (Geman et al. (1992)).

In neural networks, ensembles are often *additive* in the sense that they are defined by a weighted majority vote of network outputs (in the case of classification) or a weighted average of network outputs (in the case of regression). As we have seen, CRFs are log-linear models. As a consequence of this log-linear form, an additive ensemble of CRFs assigns a probability to a particular labelling of a graph that contains n terms, where n is the number of models in the ensemble. In general this sum

does not factorise conveniently. Consequently it is difficult to devise a Viterbi-style algorithm for efficient exact decoding with an additive ensemble of CRFs. However, if we instead use a *multiplicative* combination of CRFs we avoid such problems. The probabilities associated with a given labelling across all models then factorise into a single expression for the ensemble. We may calculate the partition function for this ensemble tractably with the same complexity as for a standard (single model) CRF. It is this ensemble model of CRFs, called a logarithmic opinion pool, that we introduce in this chapter.

This is the first of two chapters investigating LOPs for CRFs. Here we look at the simple case: combining CRF models under a LOP where the models themselves are trained independently offline before the combination step. Then, in Chapter 7, we move on to explore how we may train the models in a LOP simultaneously, with parameters in different models interacting with each other through the LOP framework.

The rest of this chapter is structured as follows: In section 6.1 we introduce logarithmic opinions pools in general and describe some of their properties, demonstrating the fact that **diversity** between the models in a LOP is important. In section 6.2 we move on to look at the specifics of how we apply LOPs to CRFs in particular. Section 6.3 investigates possible sources of diversity between the models in a LOP, and section 6.4 briefly discusses training and decoding considerations. In section 6.5 we investigate the feature set as a source of diversity, while in section 6.6 we look at diversity creation through the dataset. Having considered only unregularised models in a LOP up to this point, section 6.7 looks at LOPs created from regularised CRFs and compares the two. In section 6.8 we briefly consider the effect of non-uniform weights on the LOP. Then, in section 6.9, we investigate another possible ensemble model for CRFs, the **linear opinion pool** (LIP), and compare its properties and performance to a LOP. Finally, in section 6.10, we conclude the chapter.

6.1 Logarithmic Opinion Pools

In this section we give a general, qualitative introduction to logarithmic opinion pools. In the next section we give a more quantitative description, with precise mathematical details.

Suppose we have a set of models $\{p_\alpha\}$ where each model represents a conditional distribution[1] $p_\alpha(\mathbf{y} \mid \mathbf{x})$ over a set of random variables $\mathbf{Y}$ given another set of random

[1]Similar reasoning holds for an unconditional distribution. We use a conditional distribution here

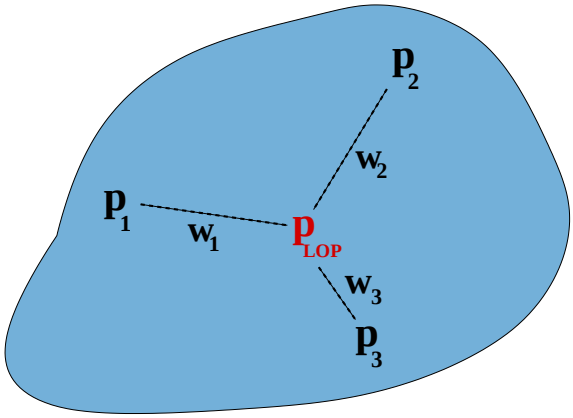

Figure 6.1: Diagrammatic representation of a LOP.

variables[2] $\mathbf{X}$. Suppose also that we have a set of weights $\{w_\alpha\}$, with weight w_α informally representing the confidence we have in the opinion represented by model p_α. Given such a set of models, a **logarithmic opinion pool** (LOP) is a single model that pools the opinions of the individual (or constituent) models. This is illustrated diagrammatically in Figure 6.1. The LOP has a distribution p_{LOP} which is defined in terms of a **weighted product** of the constituent distributions p_α, with weights w_α. Hence by changing the individual distributions p_α or the weights w_α we may change the LOP distribution p_{LOP}. The LOP is therefore an ensemble of the constituent models, but, importantly, it is a log-linear rather than linear combination of the constituents.

The weights w_α may be defined *a priori* or may be learned automatically by optimising some objective criterion. Intuitively, each weight encodes the importance we attach to the "opinion" of a particular model. For example, each distribution p_α in Figure 6.1 could represent the opinion of a particular person on a range of topics. The LOP would then represent the pooled opinions of the group, with the weights governing the importance attached to the opinions of different people.

The concept of combining the distributions of a set of models via a weighted product is not new, and has been used in a number of different application areas. Bordley (1982) derived the form for a LOP in the management science literature, applying an axiomatic approach to the problem of aggregating expert assessments of an event's probability into a group probability assessment. Benediktsson and Swain (1992) com-

because, as we will see later, our models p_α are CRFs, with conditional distributions.

[2]For convenience $\mathbf{X}$ and $\mathbf{Y}$ are assumed to be discrete, but the same reasoning here and in the next section would follow for continuous random variables, with summations replaced by the relevant integrals.

pared a number of consensus methods, including a LOP, for classification of geographic data from multiple sources, and Hansen and Krogh (2000) used a LOP of neural networks to learn protein secondary structure. However, the idea of combining CRFs under a LOP, which we present in this thesis, is new.

6.1.1 Definition

In this section we give a more quantitative description of a LOP. Given our set of constituent models $\{p_\alpha\}$ from section 6.1, and a set of associated weights $\{w_\alpha\}$, the **logarithmic opinion pool** has a distribution given by:

$$p_{\text{LOP}}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z_{\text{LOP}}(\mathbf{x})} \prod_\alpha p_\alpha(\mathbf{y} \mid \mathbf{x})^{w_\alpha} \tag{6.1}$$

with $\sum_\alpha w_\alpha = 1$, and where $Z_{\text{LOP}}(\mathbf{x})$ is the normalising function:

$$Z_{\text{LOP}}(\mathbf{x}) = \sum_{\mathbf{y}} \prod_\alpha p_\alpha(\mathbf{y} \mid \mathbf{x})^{w_\alpha} \tag{6.2}$$

In order to see how this form arises, we follow the reasoning of Heskes (1998). We start by defining the LOP as the model that is "closest" to the constituent models. For our purposes, closeness is defined in terms of Kullback-Leibler divergence (KL-divergence), which we denote by K. The KL-divergence between two conditional distributions $r_1(\mathbf{y} \mid \mathbf{x})$ and $r_2(\mathbf{y} \mid \mathbf{x})$ is given by:

$$K(r_1, r_2) = \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \sum_{\mathbf{y}} r_1(\mathbf{y} \mid \mathbf{x}) \log \left[\frac{r_1(\mathbf{y} \mid \mathbf{x})}{r_2(\mathbf{y} \mid \mathbf{x})} \right] \tag{6.3}$$

where $\tilde{p}(\mathbf{x})$ is the marginal distribution of $\mathbf{x}$. Note that, although we use KL-divergence as a distance measure between two probability distributions, it is not a metric because it is not symmetric.

The closest model p_{LOP} to the constituent models p_α is then defined in terms of a weighted KL-divergence, with real-valued weights v_α:

$$p_{\text{LOP}}(\mathbf{y} \mid \mathbf{x}) = \underset{p(\mathbf{y}|\mathbf{x})}{\operatorname{argmin}} \sum_\alpha v_\alpha K(p, p_\alpha) \tag{6.4}$$

We must ensure that the distribution over $\mathbf{y}$ for each $\mathbf{x}$ is normalised, that is:

$$\sum_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) = 1, \forall \mathbf{x} \tag{6.5}$$

In order to do this, we introduce Lagrange multipliers $\gamma_{\mathbf{x}}$. The objective becomes:

$$L = \left[\sum_{\alpha} \nu_{\alpha} \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \sum_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) \log \left[\frac{p(\mathbf{y} \mid \mathbf{x})}{p_{\alpha}(\mathbf{y} \mid \mathbf{x})} \right] \right] - \sum_{\mathbf{x}} \gamma_{\mathbf{x}} \left(\sum_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) - 1 \right) \tag{6.6}$$

where $\tilde{p}(\mathbf{x})$ is the marginal distribution of $\mathbf{x}$ (typically defined by a finite sample of data that we have).

Taking derivatives with respect to $p(\mathbf{y} \mid \mathbf{x})$ and setting to zero, we get:

$$\sum_{\alpha} \nu_{\alpha} \; \tilde{p}(\mathbf{x}) \left(\log \left[\frac{p_{\text{LOP}}(\mathbf{y} \mid \mathbf{x})}{p_{\alpha}(\mathbf{y} \mid \mathbf{x})} \right] + 1 \right) - \gamma_{\mathbf{x}} = 0 \tag{6.7}$$

Then, setting $\sum_{\alpha} \nu_{\alpha} = N$ and expanding, we get:

$$N \tilde{p}(\mathbf{x}) \log p_{\text{LOP}}(\mathbf{y} \mid \mathbf{x}) - \sum_{\alpha} \nu_{\alpha} \tilde{p}(\mathbf{x}) \log p_{\alpha}(\mathbf{y} \mid \mathbf{x}) + N \tilde{p}(\mathbf{x}) - \gamma_{\mathbf{x}} = 0 \tag{6.8}$$

and so:

$$p_{\text{LOP}}(\mathbf{y} \mid \mathbf{x}) = \exp \left(\frac{\gamma_{\mathbf{x}}}{N \tilde{p}(\mathbf{x})} - 1 \right) \exp \left(\sum_{\alpha} \frac{\nu_{\alpha}}{N} \log p_{\alpha}(\mathbf{y} \mid \mathbf{x}) \right) \tag{6.9}$$

The term $\exp \left(\frac{\gamma_{\mathbf{x}}}{N \tilde{p}(\mathbf{x})} - 1 \right)$ is dependent upon $\mathbf{x}$ but not $\mathbf{y}$, and acts as a normalising function. Denoting this by $\frac{1}{Z_{\text{LOP}}(\mathbf{x})}$, we obtain:

$$p_{\text{LOP}}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z_{\text{LOP}}(\mathbf{x})} \exp \left(\sum_{\alpha} \frac{\nu_{\alpha}}{N} \log p_{\alpha}(\mathbf{y} \mid \mathbf{x}) \right) \tag{6.10}$$

which may be re-arranged to the form given in Equation 6.1:

$$p_{\text{LOP}}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z_{\text{LOP}}(\mathbf{x})} \prod_{\alpha} p_{\alpha}(\mathbf{y} \mid \mathbf{x})^{\frac{\nu_{\alpha}}{N}} \tag{6.11}$$

From this we see that each weight w_{α} in the LOP is given by $\frac{\nu_{\alpha}}{N}$, and so the constraint $\sum_{\alpha} w_{\alpha} = 1$ falls out naturally from the derivation. Note, however, that there is

no other constraint imposed on the weights. In particular, the weights may be positive or negative. However, cases where some weights are negative are sometimes less easy to interpret than cases where all the weights are positive. To see this, consider a situation where we have two models, A and B, with probability distributions p_A and p_B respectively. These distributions encode probabilities of four outcomes, which are labelled $1 \ldots 4$. The distributions are shown in Table 6.1. From the table we can see that model A assigns a high probability of 0.97 to the first outcome but only 0.01 to each of the others. By contrast, model B considers all outcomes equally likely. Now imagine a LOP consisting only of models A and B. Table 6.2 shows the LOP's probability distributions for various weights, w_A and w_B, assigned to the two models. As we read across the table from left to right, we see increasing weight being assigned to model B in the LOP, while model A's weight decreases. In the cases where both w_A and w_B are positive, that is, in the first three cases, we obtain a probability distribution for the LOP which is in line with our intuition. For example, in the first case, where a weight of 0.99 is assigned to model A, we see that the distribution of the LOP is almost the same as that of model A. Conversely, in the third case, where the weights are reversed, the distribution of the LOP is almost the same as that of model B. For the second case, where the weights w_A and w_B are each 0.5, the LOP distribution is somewhere between the distributions for the two models. However, if we allocate a weight of greater than 1 to model B, we obtain results such as those in the fourth and fifth cases. Here the distribution of the LOP is not so clearly related to that of either model A or model B. Indeed as we continue to add more weight to model B and take weight away from model A, the LOP distribution actually looks less like that of model B. Hence by allowing negative weights, or more generally weights with magnitude greater than one, we may arrive at a situation where the resulting distribution of the LOP is less easy to interpret intuitively. In addition, when assigning weights automatically via a training algorithm, we may reduce the likelihood of overfitting by preventing the weights from taking values with large magnitude. Forcing the weights to be non-negative is one way of achieving this. As a result, from this point onwards we will impose the additional constraint that the weights be non-negative.

6.1.2 The Ambiguity Decomposition

There has been much work in the last decade investigating the properties of linear ensembles of learners. This has usually taken place within the field of neural networks,

Outcome	p_A	p_B
1	0.97	0.25
2	0.01	0.25
3	0.01	0.25
4	0.01	0.25

Table 6.1: Models A and B with probability distributions p_A and p_B respectively.

Outcome	Weights $[w_A, w_B]$				
	[0.99, 0.01]	[0.5, 0.5]	[0.01, 0.99]	$[-1, 2]$	$[-4, 5]$
1	0.9686	0.7665	0.2587	0.0034	0.0000
2	0.0105	0.0778	0.2471	0.3322	0.3333
3	0.0105	0.0778	0.2471	0.3322	0.3333
4	0.0105	0.0778	0.2471	0.3322	0.3333

Table 6.2: Probability distributions for a LOP comprising models A and B.

but the concepts apply generally to other types of learner. An ensemble is usually defined either by a weighted majority vote of the outputs of the individual learners (in the case of classification), or just a weighted average of the outputs of the individual learners (in the case of regression). Taking mean squared error as the error function, Krogh and Vedelsby (1995) demonstrate an important relationship between the generalisation error of a neural network ensemble and a property which they termed the **ambiguity** of the ensemble. This relationship may be expressed as:

$$E_{ENS} = \bar{E} - \bar{A} \tag{6.12}$$

where E_{ENS} is the generalisation error of the ensemble, $\bar{E}$ is the weighted generalisation error of the individual networks and $\bar{A}$ is the ensemble ambiguity. The ambiguity is defined as the weighted sum of the ambiguities of each network. A single network ambiguity measures the disagreement between the learner and the ensemble. This relationship is often called the **ambiguity decomposition**.

Heskes (1998) shows that a similar decomposition holds for a logarithmic opinion pool of probability distributions. In particular, suppose we have some general conditional distribution $q(\mathbf{y} \mid \mathbf{x})$. Then the following **ambiguity decomposition** holds for a

LOP of probability distributions:

$$\begin{aligned} K(q, p_{\text{LOP}}) &= \sum_{\alpha} w_\alpha K(q, p_\alpha) - \sum_{\alpha} w_\alpha K(p_{\text{LOP}}, p_\alpha) \\ &= E - A \end{aligned} \tag{6.13}$$

where, as before, the p_α are constituent models in the LOP and K denotes KL-divergence.

The principal differences between the ambiguity decompositions of Krogh and Vedelsby (in Equation 6.12) and Heskes (in Equation 6.13) are as follows:

1. In Equation 6.12 the ensemble members, for example neural networks, output scalar values representing a classification or the value of a function, whereas in Equation 6.13 the LOP members are any models representing conditional probability distributions.

2. In Equation 6.12 the outputs of the ensemble members are combined either by a weighted majority vote or a weighted linear combination, whereas in Equation 6.13 the conditional probability distributions are combined using a weighted product.

3. In Equation 6.12 the error measure is a mean squared error over the outputs of the learners, whereas in Equation 6.13 the error measure is a KL-divergence between probability distributions.

The terms E and A in Equation 6.13 are similar conceptually to their counterparts ($\bar{E}$ and $\bar{A}$) in Equation 6.12 above. The decomposition tells us that the closeness of the LOP model to the distribution $q(\mathbf{y} \mid \mathbf{x})$ is governed by a trade-off between the E and A terms. The E term represents the closeness of the individual constituent models to $q(\mathbf{y} \mid \mathbf{x})$, and the A term represents the closeness of the individual constituent models to the LOP, and therefore indirectly to each other. This latter term represents the **ambiguity** or, as we shall often refer to it, the **diversity** of the LOP. Using the decomposition, we see that in order for the LOP to be a good model of $q(\mathbf{y} \mid \mathbf{x})$, we require models $p_\alpha(\mathbf{y} \mid \mathbf{x})$ which are individually good models of $q(\mathbf{y} \mid \mathbf{x})$ (having small E) and/or diverse (having large A). In principle we can devise approaches to explicitly manipulate the E and A terms in order to create this situation. Indeed, this is what we do both later in this chapter, and in other chapters. In most cases in this thesis we tend to focus more on the A term than the E term, and look for ways of encouraging diversity.

It is interesting to consider how regularisation of the individual constituent models effects the E and A terms in the ambiguity decomposition. For example, the aim of regularising an individual constituent model would usually be to improve its generalisation capability. When regularising all constituent models, this would hopefully have the effect of decreasing the E term (as the constituent models would be closer to the "true" underlying distribution). However, regularisation would also tend to bring the constituent models closer to a given distribution. So regularising would tend to have the effect of decreasing the A term (as the constituent models are all closer to the same uniform distribution). The resulting generalisation capability of the LOP would depend on the specific interaction of these two effects. We look at this empirically later in the chapter when we apply LOPs of CRFs to some sequence labelling tasks.

An important aspect of Equation 6.13 is that the ambiguity term does not involve the distribution $q(\mathbf{y} \mid \mathbf{x})$, and can therefore be estimated using (possibly large amounts of) other data, distinct from $q(\mathbf{y} \mid \mathbf{x})$. More importantly, the other data can be *unlabelled* because the ambiguity term only involves comparing distributions over labels for the LOP and the individual models, rather than comparing either of these to some given labelling.

A consequence of this property of the ambiguity term is that an unbiased estimate of the generalisation error of the LOP may be obtained using a **cross-validation ensemble** procedure (Hansen and Krogh, 2000). With a cross-validation ensemble one has access to a labelled dataset for training and a large amount of unlabelled data. The training data is divided into n equally sized portions, where n is the number of constituent models in the LOP. Each model α is trained on all portions of the training data except portion α. The error on portion α is independent of the training and can be used to estimate the generalisation error of model α. This is done for each member α. The unlabelled data can be used to estimate the ambiguity term, as we saw above. Using the estimates of the generalisation error for each model, and the ambiguity term calculated using the unlabelled data, Equation 6.13 may be used to obtain an unbiased estimate of the generalisation error of the LOP. Most importantly, this estimate may be obtained while still utilising all of the training data for estimation, rather than requiring that part of the training data be held out in order to estimate the generalisation error.

6.1.3 Relation to Product of Experts

Hinton (1999) introduced a model similar to a logarithmic opinion pool which he called

a **Product of Experts** (PoEs). PoEs have been applied to a number of different tasks including handwriting recognition (Mayraz and Hinton, 2002), modelling character strings and the symbolic family trees problem (Brown and Hinton, 2000). Under a PoE, the probability of a particular labelling (or observation, as Hinton mainly works with generative models) is given by the product of probabilities assigned by a set of constituent models, or experts. The PoE is therefore similar in form to a LOP but has unnormalised, uniform weights set to 1:

$$p_{\text{PoE}}(\mathbf{x}) = \frac{\prod_\alpha p_\alpha(\mathbf{x})}{\sum_{\mathbf{x}'} \prod_\alpha p_\alpha(\mathbf{x}')} \tag{6.14}$$

Given a set of observed data vectors, Hinton attempts to fit the PoE model by maximising the log-likelihood of the data under the model. Following his notation, and denoting the set of parameters in model α by θ_α, the derivative of the log-likelihood of a single observed data vector $\mathbf{x}$ with respect to the parameters in model α is:

$$\begin{aligned} \frac{\partial \log p_{\text{PoE}}(\mathbf{x} \mid \theta_1, \ldots, \theta_n)}{\partial \theta_\alpha} &= \frac{\partial \log p_\alpha(\mathbf{x} \mid \theta_\alpha)}{\partial \theta_\alpha} \\ &- \sum_{\mathbf{x}'} p_{\text{PoE}}(\mathbf{x}' \mid \theta_1, \ldots, \theta_n) \frac{\partial \log p_\alpha(\mathbf{x}' \mid \theta_\alpha)}{\partial \theta_\alpha} \end{aligned} \tag{6.15}$$

Hinton works with models where although the calculation of the derivative of the log-likelihood for a single expert (first term in Equation 6.15) is tractable, the calculation of the expected value in the second term is intractable because of the summation over the space of all data vectors. This problem may be tackled in one of two ways:

- Use some form of sampling, for example Gibbs sampling, to sample from the distribution $p_{\text{PoE}}(\mathbf{x}' \mid \theta_1, \ldots, \theta_n)$ in the second term of Equation 6.15, and so approximate the value of the second term.
- Employ a different parameter estimation procedure, with a different objective function.

Hinton (2002) takes the second alternative above, and introduces a parameter estimation technique that he calls **contrastive estimation**. Note that, as we see later, our models are linear chain CRFs and so we can efficiently evaluate the equivalent of the second term in Equation 6.15 *exactly*, and do not need to resort to approximate methods. We therefore do not use contrastive estimation in this thesis.

6.2 LOPs for CRFs

Up to this point our discussion of LOPs has been very general, without regard to the kind of models providing the distributions p_α. From here on we make the discussion more concrete, and consider LOPs of CRFs. We could apply the ideas we develop to CRFs with any graphical structure, such as chains, trees or lattices. For illustration, we use the sequential labelling tasks we described in Chapter 3. We therefore work with CRFs with a linear chain structure. In this case, the variables $\mathbf{x}$ represent an observation sequence, for example a sequence of words in a sentence. The variables $\mathbf{y}$ represent a sequence of labels for the observation sequence. For example, in the case of named entity recognition (NER)[3], these would be NER labels.

To clarify the terminology that we will use from here onwards, note the distinction between the *weights* w_α (sometimes referred to as *per-model weights*) used in the weighted product in the LOP (appearing explicitly in Equation 6.1), and the *parameters* $\lambda_{\alpha k}$ which parameterise each constituent CRF α (appearing implicitly in Equation 6.1 through the p_α).

As we alluded to in the preamble to this chapter, because CRFs are log-linear models they are particularly well suited to combination under a LOP. To see this, consider again Equation 6.1 and the form for a CRF distribution discussed in Chapter 2. It is easy to see that when CRF distributions are combined under a weighted product, the potential functions factorise so that the resulting distribution is itself that of a CRF. This CRF has potential functions that are given by a weighted log-linear combination of the potential functions of the constituent models, with weights w_α.

It is clear from the general form of the LOP that if any constituent model allocates zero probability to a labelling $\mathbf{y}$ then the LOP's probability for that labelling also collapses to zero. Hence the opinion of each constituent model can act as a veto. This is a potential weakness of the LOP form, but is not a problem for us because CRFs, being exponential models, generally do not allocate zero probability to any labelling.

6.3 Sources of Diversity

In section 6.1.2, we saw how the ambiguity decomposition motivates the desire to construct constituent models p_α for a LOP that are both individually good models of a distribution and are diverse. Diversity between the constituent models may be created

[3] See Chapter 3 for a description of this task.

in a number of different ways. The main strategies for introducing diversity explored here are:

(1) **Feature set.** Constituent models are parameterised using different feature sets. The feature sets may be created in one of two ways:

- *Manually*: feature sets are defined using human intuition about which sets are likely to lead to models with diverse distributions. This is usually based on feature sets providing alternative, diverse "views" on the data.
- *Automatically*: feature sets are generated using a feature induction/clustering process by optimising some objective which tries to maximise diversity between the models as they grow.

In this thesis we explore only the *manual* creation of feature sets above. The *automatic* creation of feature sets is a much harder problem and is beyond the scope of the thesis.

(2) **Training set.** Constituent models are trained using different training sets. The variation in the properties of the training sets creates diversity between the models that are created using them. The different training sets may be created in a number of ways. Examples include:

- *Partitioning*: the training set is partitioned into subsets with each constituent model being trained on a separate subset.
- *Bagging*: each constituent model is trained on a set re-sampled from the original training set distribution.
- *Boosting*: training set instances are given different weights during training on the basis of how difficult they are to label.

In this thesis we explore *bagging* as a representative example of diversity via the training set.

(3) **Training algorithm.** The training algorithm is designed so that, in addition to modelling the training set well, the constituent models are encouraged or forced to be diverse from one another. This means that the models are coupled during the training process and the parameters in all the models are trained together. We call this **co-operative training**, and investigate it Chapter 7. As we will see in that

chapter, we introduce a penalty term in the objective function that explicitly tries to maximise the ambiguity term for the LOP.

Note that the three approaches above are not mutually exclusive, and could be undertaken in combination. However, for ease of exposition we will deal with them separately. In this chapter we look at cases (1) and (2), contrasting them as possible sources of constituent model diversity. Case (3), the co-operative training, is a little more involved and so is covered separately in Chapter 7.

Consistent with the Product of Experts model discussed in section 6.1.3, we will often refer to the constituent models of a LOP of CRFs as **experts**. Sometimes an expert will focus on modelling a particular aspect or subset of a probability distribution well (hence the name). An example of this would be an expert that consists almost entirely of features that fire for a particular label, thereby modelling the distribution of that label while effectively ignoring the details of the distributions of other labels. At other times an expert may model the entire distribution but with an alternative "view" to another expert. An example here would be two experts which model the distribution of all labels, but which consist of different feature sets. Sometimes we will also use the name **constituent model** for **expert** (as we have been doing so far). Therefore, for the rest of the thesis, we will treat the two terms as interchangeable.

6.4 Training and Decoding

In this chapter *training* simply refers to the process of estimating the parameters of the constituent models. Each constituent model is trained independently of the others, offline. The constituent models are then combined under a LOP with uniform weights, i.e. each model being weighted equally. As we saw in section 6.2, a LOP of CRFs is itself a CRF. Consequently, decoding the LOP is no more computationally complex than decoding with a standard CRF.

6.5 Diversity via the Feature Set

In this section we focus on trying to manually create diverse experts by manipulating the feature set that each expert uses. To do this we first define a pool of features and then create different partitions of this pool. For the pool we use the feature set of the STANDARD CRF, described in Chapter 4. Each partition of the feature set defines

a set of experts. We call such a set an **expert set**. The expert sets we use for our experiments are described below.

6.5.1 Expert Sets

The expert sets we use to build LOPs are defined via our intuition about which groupings of features should result in a set of accurate and diverse experts. Usually these experts are designed to focus on modelling a particular aspect or subset of the distribution. As we saw earlier, the aim here is to define experts that model parts of the distribution well while retaining diversity. In this section we describe four expert sets. We include the STANDARD CRF model in each expert set, as one of the experts.

6.5.1.1 Label

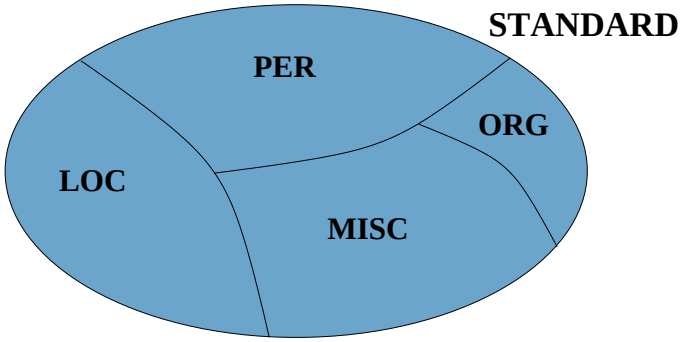

Figure 6.2: LABEL expert set for NER.

The **LABEL** expert set consists of the STANDARD CRF and a partition of the features in the STANDARD CRF into five experts, one for each label. For NER an expert corresponding to label X consists only of features that involve labels B-X or I-X at the current or previous positions, while for POS tagging an expert corresponding to label X consists only of features that involve label X at the current or previous positions. This situation is illustrated in Figure 6.2 for the NER case. Here the shaded oval, representing all features in the STANDARD model, is partitioned into subsets corresponding to each NER label. In this expert set the experts focus on trying to model the distribution of a particular label.

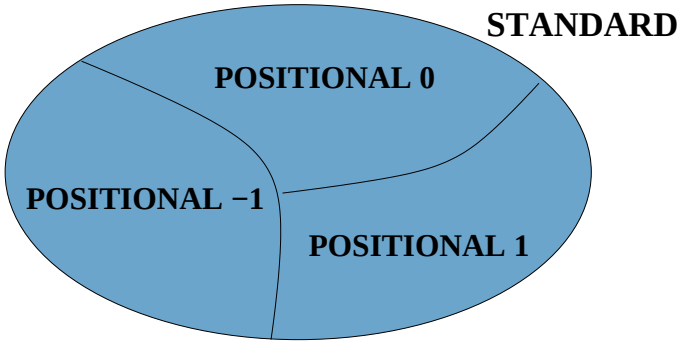

Figure 6.3: POSITIONAL expert set.

6.5.1.2 Positional

The **POSITIONAL** expert set consists of the STANDARD CRF and a partition of the features in the STANDARD CRF into three experts, each consisting only of features that involve events either behind, at or ahead of the current sequence position. This is illustrated in Figure 6.3. The experts in this expert set focus on trying to model the dependence of the current prediction on different positional information.

6.5.1.3 Simple

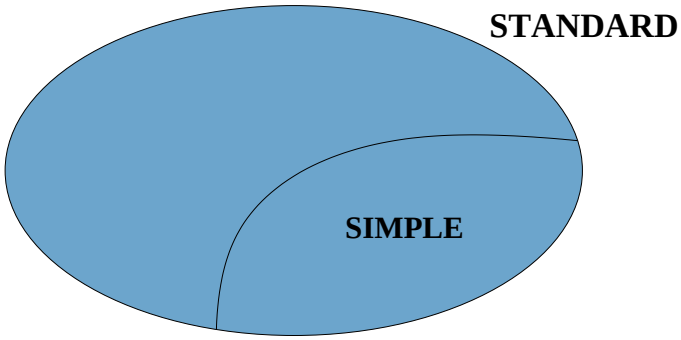

Figure 6.4: SIMPLE expert set.

The **SIMPLE** expert set consists of the STANDARD CRF and just a single expert: the SIMPLE CRF described in Chapter 4 (one of the reference models from that chapter). This situation is illustrated in Figure 6.4. The SIMPLE CRF models the entire distribution rather than focusing on a particular aspect or subset, but is much less expressive than the STANDARD model. The SIMPLE model comprises $24,819$ features for NER and $18,482$ features for POS tagging.

6.5.1.4 Random

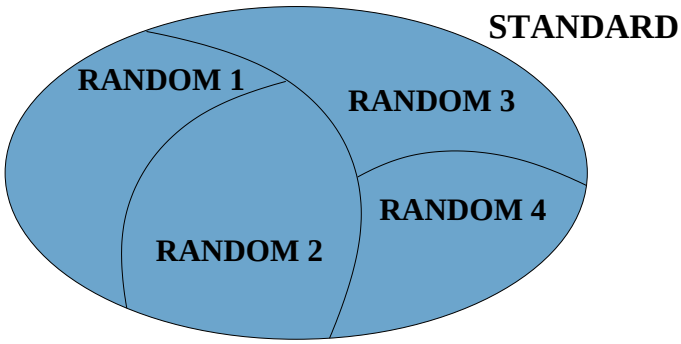

Figure 6.5: RANDOM expert set.

The **RANDOM** expert set consists of the STANDARD CRF and random partitions of the features in the STANDARD CRF into four experts. This is illustrated in Figure 6.5. This expert set acts as a baseline to ascertain the performance that can be expected from an expert set that is not defined via any linguistic intuition.

6.5.2 Results

6.5.2.1 Experts

Before presenting results for the LOPs, we briefly give performance figures for the STANDARD CRF and expert CRFs in isolation. For illustration, we do this for NER models only. Table 6.3 shows F scores on the development set for the NER experts. At the top we have the STANDARD CRF, consisting of the entire feature set, both unregularised and regularised with a Gaussian prior. We then have the individual experts from each expert set presented in section 6.5.1. We see that, as expected, the expert CRFs in isolation model the data relatively poorly compared to the STANDARD CRF. For example, some of the label experts attain very low F scores as they focus only on modelling one particular label. Similar behaviour was observed for the POS tagging models.

6.5.2.2 LOPs

Having defined and trained the experts above, we combine the experts from a given expert set under a LOP with uniform weights. Table 6.4 gives F scores for the resulting

Expert	F score
STANDARD unreg.	88.21
STANDARD reg.	89.86
LABEL LOC	8.82
LABEL MISC	8.38
LABEL ORG	9.47
LABEL PER	11.86
LABEL O	58.52
POSITIONAL -1	73.07
POSITIONAL 0	86.98
POSITIONAL 1	73.15
SIMPLE	79.53
RANDOM 1	71.19
RANDOM 2	69.11
RANDOM 3	73.29
RANDOM 4	69.30

Table 6.3: Development set F scores for individual NER experts.

LOPs on NER. Scores for both unregularised and regularised versions of the STANDARD CRF are included for comparison. From the table we observe the following points for NER:

1. In every case except one the LOPs outperform the unregularised STANDARD CRF on both the development and test sets at a significance level of $p < 0.05$. The one exception is the LOP with RANDOM experts on the development set, which, although obtaining a higher F score than the unregularised STANDARD CRF, does not outperform it at the $p < 0.05$ level. However, as we discussed in section 6.5.1.4, the RANDOM expert set is only intended as a baseline.

2. In every case except the LOP with RANDOM experts on the development set, the LOPs do not significantly underperform the *regularised* STANDARD CRF at a significance level of $p < 0.05$. Indeed, in the case of the SIMPLE expert set, LOPs on both the development and test sets outperform the regularised STANDARD CRF.

Our results for NER therefore show that uniformly-weighted LOPs with unregularised experts can lead to performance improvements over an unregularised STANDARD CRF that equal or exceed those achieved from a conventional regularisation approach using a Gaussian prior.

The results for LOPs on POS tagging show the same general trends as those on NER, but with some of the above characteristics being slightly less marked. Table 6.5 gives accuracies on POS tagging for the LOPs corresponding to those in Table 6.4 on NER. As before, scores for both unregularised and regularised versions of the STANDARD CRF are included for comparison. In every case except the RANDOM expert set on the development set, the LOPs significantly outperform the unregularised STANDARD CRF. In addition, the LOP with the SIMPLE expert set significantly outperforms the regularised STANDARD CRF on the development set, and none of the LOPs on the test set are significantly worse than the regularised STANDARD CRF. The POS tagging results therefore support our findings on NER.

In addition to providing a competitive alternative to conventional regularisation with a prior, the LOP approach, when used with unregularised experts, has the added advantage that it is "parameter-free". By this we mean that each expert CRF in the LOP is unregularised, so we are not required to search a hyperparameter space. As an illustration, to obtain our best results for the POS tagging regularised STANDARD model, we re-trained using 15 different values of the Gaussian prior variance. With the LOP we trained each expert CRF only *once*.

Another important difference between the LOP approach presented here and that of conventional regularisation with a prior is that the experts that comprise the LOP are generally small, compact models that are fast to train. In the case of conventional regularisation, however, it is the STANDARD model that must be re-trained many times for different hyperparameter values. We conclude, then, that using experts defined by intuitively-motivated feature partitions in a uniformly-weighted LOP can provide a competitive alternative to conventional regularisation using a prior, and is much cheaper computationally.

6.5.3 Choice of Expert Sets

We can see from Tables 6.4 and 6.5 that the performance of a LOP of CRFs varies with the choice of expert set. For example, in our tasks the SIMPLE and POSITIONAL expert sets perform better than those for the LABEL and RANDOM sets. For an ex-

Model/LOP	Development set	Test set
STANDARD unreg.	88.21	81.60
STANDARD reg.	89.86	83.97
LABEL	89.23	83.55
POSITIONAL	89.86	84.50
SIMPLE	90.06	84.03
RANDOM	88.77	83.13

Table 6.4: F scores for LOPs of uniformly-weighted expert sets on NER.

Model/LOP	Development set	Test set
STANDARD unreg.	96.76	96.32
STANDARD reg.	97.07	96.69
LABEL	96.86	96.62
POSITIONAL	96.91	96.65
SIMPLE	97.30	97.09
RANDOM	96.81	96.66

Table 6.5: Accuracies for LOPs of uniformly-weighted expert sets on POS tagging.

planation here, we refer back to our discussion of Equation 6.13. We conjecture that the SIMPLE and POSITIONAL expert sets achieve good performance in the LOP because they consist of experts that are diverse while simultaneously being reasonable models of the data. The LABEL expert set exhibits greater diversity between the experts, because each expert focuses on modelling a particular label only, but each expert is a relatively poor model of the entire distribution and the corresponding LOP performs worse. Similarly, the RANDOM experts are in general better models of the entire distribution than the LABEL experts but tend to be less diverse because they do not focus on any one aspect or subset of it. Intuitively, then, we want to devise experts that provide diverse but accurate views on the data.

The expert sets we present in this chapter are motivated by linguistic intuition, but clearly many other choices exist. One possibility would be to develop a framework that establishes the feature partitions automatically, using a feature induction process across all the constituent models simultaneously. We discuss this briefly in the future

work section of Chapter 9.

6.6 Diversity via the Dataset

Having investigated the feature set as a possible source of diversity in the previous section, in this section we consider the training data as the source. To do this we create a set of constituent models using a fixed set of feature templates (the same set as earlier) but instantiate these templates on varying training datasets. To create the extra datasets we *bag* the training data. Specifically, we randomly select sequences from the training data, with replacement, to create a new dataset of the same size as the training data. We do this 15 times in total for each task, to create a set of bagged training datasets. We then use the feature templates of the STANDARD CRF model to instantiate and extract features on each of the bagged training datasets. Clearly, each bagged dataset is a subset of the original training dataset. Therefore, each feature set derived from a bagged dataset will be a subset of the feature set of the STANDARD model. For NER, the STANDARD model has $450,346$ features, whereas the feature sets from the bagged training datasets have sizes ranging from $331,572$ to $336,871$ features, with a mean size of $334,039$. Similarly, for POS tagging the STANDARD model has $189,339$ features, whereas the feature sets from the bagged training datasets have sizes ranging from $144,219$ to $147,838$ features, with a mean size of $145,796$.

Having created the feature sets, we train the corresponding models without regularisation. Clearly, as the bagged models are trained on less data than the STANDARD model for each task, we would expect them to individually underperform the STANDARD model. To illustrate, on NER the unregularised STANDARD model obtains an development set F score of 88.21, whereas the bagged models obtain development set F scores ranging from 84.62 to 85.48, with a mean score of 85.04. A similar pattern is observed for POS tagging.

For each task, we then combine the bagged models under a LOP and decode the development and test sets. We would like to see whether the number of constituent models in the LOP affects performance, so we create LOPs of size 3, 5 and 15. For the LOPs of size 3 and 5 we create several constituent model sets from randomly selected bagged models and average the results. For the LOP of size 15, we clearly only have a single constituent model set, containing all the bagged models.

Table 6.6 gives F scores on NER for uniformly-weighted LOPs constructed from bagged CRF models. Table 6.7 gives accuracies for the corresponding LOPs on POS

tagging. In each case we include scores for the unregularised and regularised versions of the STANDARD model. From the tables we can see that:

1. As the number of constituent models in the LOP increases, scores generally improve. For LOPs of size 3 and 5, the results are little better than the unregularised STANDARD CRF in most cases, and worse in some cases.

2. The results for the LOP of size 15 are closer to those of the LOPs of the last section, where we were using feature set experts. In three of the four cases, the LOP of size 15 significantly outperforms the unregularised STANDARD CRF at the $p < 0.05$ significance level, and in two of the four cases does not significantly underperform the regularised STANDARD CRF.

These results generally show that using the training data as a source of diversity for the creation of constituent models is not as effective as using the feature set. The bagged model LOPs do provide some improvement over the unregularised STANDARD model in some cases, but the results are generally not competitive with those of the regularised STANDARD model.

From the point of view of diversity, these results are to be expected. The feature set experts of the previous section are intuitively more diverse than the constituent models in this section. For example, the LABEL experts each focus on modelling a different aspect of label distribution, and have very little overlap of their feature sets. In contrast, as we saw above in this section, the bagged CRF models have significant feature set overlap with the STANDARD model, and therefore with each other. Note that bagging the training set and partitioning the feature set are essentially orthogonal approaches for creating constituent models for LOPs, so it would be possible to develop hybrid schemes that combine the two.

There are other disadvantages with using bagged models in LOPs, as compared to feature set experts. Firstly, it appears that LOPs of bagged models only attain good performance when a large number of constituent models are included. This means total training time is longer than with the smaller expert sets of the previous section. Secondly, each bagged model is relatively large, containing a substantial proportion of the features in the STANDARD model. This makes the bagged models slow to train. The feature set experts, by contrast, are generally quite small models and so are faster to train. These factors contribute to our conclusion that using the training data as a source of diversity for creating constituent models for LOPs is of limited use.

Model/LOP	Development	Test
STANDARD unreg.	88.21	81.60
STANDARD reg.	89.86	83.97
Mean 3	87.05	81.78
Mean 5	87.57	82.29
Mean 15	88.05	83.14

Table 6.6: F scores for LOPs of uniformly-weighted bagged models on NER.

Model/LOP	Development	Test
STANDARD unreg.	96.76	96.32
STANDARD reg.	97.07	96.69
Mean 3	96.72	96.43
Mean 5	96.83	96.52
Mean 15	96.95	96.64

Table 6.7: Accuracies for LOPs of uniformly-weighted bagged models on POS tagging.

Consequently, from this point onwards in the thesis we will focus on LOPs containing feature set experts.

6.7 LOPs with Regularised Experts

All of the results presented so far in this chapter have involved LOPs with unregularised constituent models. In this section we relax this constraint and consider the effect on LOP performance of using regularised experts. We regularise the experts using a Gaussian prior over the model parameters, the Gaussian being representative of the different priors we considered in Chapter 5.

Table 6.8 shows F scores on NER for LOPs created from regularised versions of the feature set experts described in section 6.5. We include the results for the unregularised models for comparison. Table 6.9 gives accuracies for the corresponding models on POS tagging. The results show that regularising the experts has a mixed effect on LOP performance, with no consistent pattern emerging. In some cases scores improve, in other cases they worsen. Viewing this from a diversity angle, this is not surprising.

LOP	Unregularised		Regularised	
	Development	Test	Development	Test
LABEL	89.23	83.55	89.68	84.10
POSITIONAL	89.86	84.50	89.21	82.84
SIMPLE	90.06	84.03	90.00	84.89
RANDOM	88.77	83.13	88.65	83.01

Table 6.8: F scores for LOPs with uniformly-weighted unregularised and regularised expert sets on NER.

LOP	Unregularised		Regularised	
	Development	Test	Development	Test
LABEL	96.86	96.62	96.66	96.39
POSITIONAL	96.91	96.65	96.78	96.61
SIMPLE	96.30	97.09	97.34	97.23
RANDOM	96.81	96.66	96.16	96.08

Table 6.9: Accuracies for LOPs with uniformly-weighted unregularised and regularised expert sets on POS tagging.

When regularising the constituent models, usually the generalisation capability of the individual models will improve, but the models will also tend to drawn closer to the uniform distribution, and therefore to each other. Hence there will be a trade-off between individual constituent model accuracy and diversity between the models. The result of this trade-off is difficult to predict, and will vary with the LOP. Of course, using regularised experts in a LOP negates the "parameter-free" aspect that we earlier proposed as a strength of the LOP with unregularised experts.

6.8 Non-Uniform Weights

So far in this chapter we have only considered LOPs with uniform weights. We have seen that particularly good results may be obtained by combining unregularised feature set experts in a LOP where all models are equally weighted. However, it is clearly possible to employ non-uniform weight combinations in a LOP. Doing so gives pref-

LOP	Development set	Test set
SIMPLE uniform	90.06	84.03
SIMPLE manual	90.07	84.54

Table 6.10: F scores for manually-weighted SIMPLE LOPs on NER.

LOP	Development set	Test set
SIMPLE uniform	97.30	97.09
SIMPLE manual	97.34	97.18

Table 6.11: Accuracies for manually-weighted SIMPLE LOPs on POS tagging.

erence to the opinion of some models over others. To find a suitable set of weights for a given set of constituent models in a LOP, we may generally use one of methods: *manual* adjustment or *automatic* search.

Efficient manual weight adjustment is difficult for LOPs that contain more than two constituent models as these LOPs have at least two *independent* weights[4]. Searching for an optimal set of weights in this situation is akin to the problem of searching for optimal hyperparameters for a multi-dimensional prior, that we saw in Chapter 5. However, LOPs of size two only have one independent weight, which may easily be optimised using a development set. We do this for the SIMPLE LOPs with unregularised experts we saw in section 6.5.2. Tables 6.10 and 6.11 show F scores on NER and accuracies on POS tagging, respectively, for these SIMPLE LOPs with manually adjusted weights. We can see that on both tasks improvements can be made over the uniformly-weighted LOPs by manually adjusting the weights away from the uniform distribution. Development set scores for the manually-adjusted LOPs are higher than those of the uniform LOPs, but not significantly so at the $p < 0.05$ level. However, on both tasks test set scores *are* significantly higher.

As for automatic determination of optimal weight combinations, different possibilities exist. One approach is to formulate a relevant objective function and optimise it. One candidate here is the log-likelihood of the LOP as a function of the weights. In this setup we fix the constituent models in their pre-defined state and train the weights in the LOP as a separate process. To do this we must evaluate the derivatives of the

[4]Remember we are working with normalised weights.

log-likelihood with respect to the weights. This is relatively straight forward, with the derivatives taking a form that is similar to the derivatives for a standard CRF with respect to its parameters. In particular, the expected value of every feature in each constituent model under the LOP distribution must be evaluated. This can be achieved efficiently using the standard forward-backward algorithm because, as we noted earlier, the LOP is itself a CRF. We employed this weight-training procedure for the feature set experts described in section 6.5. We found that the approach led to minimal improvements in performance over the corresponding LOPs with uniform weights. One possible reason for this is that in general the feature set expert sets we have been using contain constituent models that are of roughly equal quality (with the possible exception of the SIMPLE expert set). For these expert sets the best weight combinations probably do not lie very far from the uniform distribution, so any automatic process will struggle to significantly improve on the uniform LOPs. It is possible, therefore, that this automatic procedure may perform better for LOPs that contain a greater differential in the quality of the constituent models. We leave this possibility as an avenue for further work.

6.9 Linear Opinion Pools

At the beginning of this chapter we saw how a LOP is a natural candidate for an ensemble model of CRFs due to the exponential form of the CRF distribution: a set of CRFs combined under a LOP is itself another CRF. Clearly, if we combine CRFs using an additive ensemble rather than a multiplicative one, we do not obtain such an elegant compact representation. However, it is still interesting to ask the question as whether there is an efficient way to decode a additive model of CRFs, and how such a model would perform in comparison to a LOP. In this section we consider these issues.

In contrast to a LOP, an additive model of classifiers is called a **linear opinion pool (LIP)**. Using our notation for CRF distributions p_α and per-model weights w_α from earlier in this chapter, we define a LIP of CRFs as:

$$p_{\text{LIP}}(\mathbf{y} \mid \mathbf{x}) = \sum_\alpha w_\alpha p_\alpha(\mathbf{y} \mid \mathbf{x}) \tag{6.16}$$

with $\sum_\alpha w_\alpha = 1$. Unlike a LOP, with a LIP there is no requirement for a normalising function because the right-hand side of Equation 6.16 is already normalised. To see this, we sum over all sequences $\mathbf{y}$:

$$\begin{aligned}\sum_{\mathbf{y}} p_{\text{LIP}}(\mathbf{y}\,|\,\mathbf{x}) &= \sum_{\mathbf{y}}\sum_{\alpha} w_\alpha p_\alpha(\mathbf{y}\,|\,\mathbf{x}) \\ &= \sum_{\alpha} w_\alpha \sum_{\mathbf{y}} p_\alpha(\mathbf{y}\,|\,\mathbf{x}) \\ &= \sum_{\alpha} w_\alpha \\ &= 1 \end{aligned} \tag{6.17}$$

Our question, then, is whether we can efficiently decode a LIP using a Viterbi algorithm. We want to find the Viterbi path $\mathbf{y}^*$ where $\mathbf{y}^* = \text{argmax}_{\mathbf{y}}\, p_{\text{LIP}}(\mathbf{y}\,|\,\mathbf{x})$ for a given observation sequence $\mathbf{x}$. Let us first expand the definition of the Viterbi path here:

$$\begin{aligned}\underset{\mathbf{y}}{\text{argmax}}\, p_{\text{LIP}}(\mathbf{y}\,|\,\mathbf{x}) &= \underset{\mathbf{y}}{\text{argmax}} \left[\sum_{\alpha} w_\alpha p_\alpha(\mathbf{y}\,|\,\mathbf{x})\right] \\ &= \underset{\mathbf{y}}{\text{argmax}} \left[\sum_{\alpha} \frac{w_\alpha}{Z(\mathbf{x})} \exp\left(\sum_{t}\sum_{k} f_{\alpha k}(y_{t-1}, y_t, t)\right)\right] \end{aligned} \tag{6.18}$$

where t runs over the cliques in a sequence, k runs over the features in a model and $f_{\alpha k}$ is the kth feature in model α. It is tempting to define a Viterbi-style algorithm that moves left to right through a lattice maximising the partial sum (i.e. up to a certain point in the sequence) on the right-hand side of Equation 6.18. However, a little thought shows that this would not necessarily recover the Viterbi path of the LIP. The reason is that the partial sums contain co-efficients $\frac{w_\alpha}{Z(\mathbf{x})}$ that are "wrong" except on the last clique of the sequence. This is because the partition functions $Z(\mathbf{x})$ for each sequence cover the whole sequence, rather than the partial sequence to a particular point. As a result, it is difficult to see a simple way to decode the LIP using a Viterbi algorithm.

One solution to this problem involves making a approximation. Rather than seeking the Viterbi path of the LIP, we may instead look for the path that maximises the product of the LIP's marginal label probabilities at each point in the sequence. We make the assumption that this latter path is a good approximation for the Viterbi path. Indeed, in many cases the two paths will coincide. We are assuming, then, that:

$$p_{\text{LIP}}\left(\mathbf{y} = \mathbf{y}'\,|\,\mathbf{x}\right) \approx \prod_{t} p_{\text{LIP}}\left(y_t = y'_t\,|\,\mathbf{x}\right) \tag{6.19}$$

But we know that:

$$\begin{aligned} p_{\text{LIP}}\left(y_t = y'_t \mid \mathbf{x}\right) &= \sum_{\mathbf{y}: y_t = y'_t} p_{\text{LIP}}\left(\mathbf{y} \mid \mathbf{x}\right) \\ &= \sum_{\mathbf{y}: y_t = y'_t} \sum_{\alpha} w_\alpha p_\alpha\left(\mathbf{y} \mid \mathbf{x}\right) \\ &= \sum_{\alpha} w_\alpha \sum_{\mathbf{y}: y_t = y'_t} p_\alpha\left(\mathbf{y} \mid \mathbf{x}\right) \\ &= \sum_{\alpha} w_\alpha p_\alpha\left(y_t = y'_t \mid \mathbf{x}\right) \end{aligned} \tag{6.20}$$

Hence the LIP's marginal label distribution at given point is equal to the weighted sum of the marginal label distributions of the constituent models at that point, using the weights w_α. Therefore, using Equation 6.19:

$$\begin{aligned} \operatorname*{argmax}_{\mathbf{y}} p_{\text{LIP}}\left(\mathbf{y} \mid \mathbf{x}\right) &= \operatorname*{argmax}_{\mathbf{y}} \left[\prod_t \sum_{\alpha} w_\alpha p_\alpha\left(y_t \mid \mathbf{x}\right)\right] \\ &= \prod_t \left[\operatorname*{argmax}_{y_t} \left[\sum_{\alpha} w_\alpha p_\alpha\left(y_t \mid \mathbf{x}\right)\right]\right] \end{aligned} \tag{6.21}$$

From this we can see that in order to decode a sequence with observation $\mathbf{x}$ we need to do the following:

1. Calculate the pointwise marginal label distributions $p_\alpha\left(y_t \mid \mathbf{x}\right)$ for each model α at each point t. This can be achieved by conducting a backward sweep through the decoding lattice, and is explained in Chapter 2.

2. Take a weighted sum of these marginal distributions at each point t using the weights w_α.

3. For each point t select the label that corresponds to the highest probability in the weighted marginal distribution at that point.

In order to compare the performance of LIPs and LOPs, we use the procedure described above to decode a set of uniformly-weighted LIPs. Table 6.12 gives F scores on NER for LIPs that consist of the feature set experts described in section 6.5. The F scores for the corresponding LOPs are included for comparison. Table 6.13 gives similar data on POS tagging, with accuracies replacing F scores. From the tables we see that for both tasks, in every case except one, the scores obtained by the LOPs are higher than those of the LIPs on both the development and test sets. For NER, of the eight pairwise comparisons, five of them represent a significant performance

Opinion Pool	LOP		LIP	
	Development	Test	Development	Test
LABEL	89.23	83.55	88.59	82.24
POSITIONAL	89.86	84.50	88.73	82.32
SIMPLE	90.06	84.03	89.33	82.99
RANDOM	88.77	83.13	87.28	81.23

Table 6.12: F scores for LOPs and LIPs with uniformly-weighted expert sets on NER.

improvement of the LOP over the corresponding LIP at the $p < 0.05$ significance level. Equally, for POS tagging, six of the eight pairwise comparisons result in a significant improvement for the LOPs.

The two primary factors responsible for the differences in performance between the LOPs and the LIPs are likely to be the combination method (additive versus multiplicative) and the approximation method (as compared to an exact decoding). It is not clear which of these factors dominates. In a sense, though, this is unimportant because even with the approximation the LIP is significantly more complex to decode than the LOP. With a LIP, using the method above, each constituent model must decode separately, including both a forward and backward sweep through the lattice. With a LOP, by contrast, the constituent models are combined upfront and only one decoding takes place, with only the forward sweep through the lattice needed to recover the Viterbi path. Therefore the LOP seems to be by far the more preferable choice. As a result, we will not be considering LIPs beyond this point in the thesis.

Note that Sutton et al. (2006) also consider linear ensembles of CRFs, looking at both sequential and pointwise combinations. They reach similar conclusions to the ones we reach here.

6.10 Summary

Following our examination of conventional regularisation techniques for CRFs in Chapter 5, in this chapter we moved on to introduce an alternative framework for CRF regularisation based on a form of ensemble model. The model, called a **logarithmic opinion pool**, combines a set of CRFs in a weighted product. The LOP is a natural choice for a CRF ensemble due to the exponential form of the CRF distribution. We

Opinion Pool	LOP		LIP	
	Development	Test	Development	Test
LABEL	96.86	96.62	96.52	96.45
POSITIONAL	96.91	96.65	96.93	96.60
SIMPLE	97.30	97.09	97.23	97.01
RANDOM	96.81	96.66	96.64	96.31

Table 6.13: Accuracies for LOPs and LIPs with uniformly-weighted expert sets on POS tagging.

described the properties of a LOP, and, in particular, saw how a LOP satisfies an ambiguity decomposition, which motivates the need for the constituent models in a LOP to be accurate and/or diverse. We discussed a number of ways in which such diverse models may be created, focusing in this chapter on the feature set and the training data as possibles sources of diversity.

Our main finding comes from the use of the feature set as a source of diversity. By creating a set of expert CRF models based on an intuitively-motivated partition of the feature set, and combining them under a LOP with uniform weights, we may obtain a model which significantly outperforms an unregularised STANDARD CRF that utilises the entire feature set, and is comparable in performance to a STANDARD CRF regularised with a Gaussian prior. This means that the LOP approach represents a competitive alternative to conventional regularisation with a prior, but without the need to search a (possibly high-dimensional) hyperparameter space.

Using the training set as a source of diversity, we created LOPs from a set of bagged CRF constituent models. Our results show that although bagged model LOPs do provide some improvement over an unregularised STANDARD model, the results are not competitive with those of the regularised STANDARD model. In addition, the bagged constituent models are generally slower to train than the feature set constituent models. We conclude that LOPs built from feature set experts are preferable.

Our main claim regarding the competitiveness of LOPs as an approach to regularising CRFs relies on the use of unregularised models in the LOP. However, in the chapter we also considered regularised CRF models as LOP constituents. We found that results were mixed relative to the corresponding LOPs with unregularised constituents, and were difficult to predict due the subtle trade-off between model accuracy

and diversity.

Finally, we compared a linear opinion pool to a LOP. We saw that it is difficult to devise an efficient Viterbi-style algorithm for LIP decoding, and that we must therefore resort to an approximate decoding. However, making this approximation leads to decoding that is both more expensive than LOP decoding and results in lower performance. We therefore conclude that a LOP is a much more preferable ensemble model for CRFs than a LIP.

In this chapter constituent models were trained independently, offline, then combined under a LOP. In the next chapter we look at how we may train the constituent models non-independently, with parameters in different models interacting during training in such a way as to encourage diversity between the resulting models.

Chapter 7

Co-operative Training of LOPs

In Chapter 6 we introduced the idea of a **logarithmic opinion pool** (LOP) of CRFs. The pool comprised a product of individual CRF models weighted by a set of per-model weights. We also saw how a logarithmic opinion pool and its constituent models satisfy an **ambiguity decomposition**:

$$K(q, p_{\text{LOP}}) \quad = \quad E - A = \sum_{\alpha} w_{\alpha} K(q, p_{\alpha}) - \sum_{\alpha} w_{\alpha} K(p_{\text{LOP}}, p_{\alpha}) \tag{7.1}$$

where p_{LOP} is the LOP, the p_{α} are the constituent models, q is some general distribution and the w_{α} are the LOP per-model weights. As we described in some detail in section 6.1.2, the E term represents the closeness of the individual constituent models to q and the A term represents the **ambiguity** or **diversity** between models, i.e. the closeness of the individual constituent models to the LOP and therefore to each other. We can see from the decomposition that in order for the LOP to be a good model of q, we require models p_{α} which are both individually good models of q (having small E) and/or diverse (having large A). In Chapter 6 we explored different ways of encouraging diversity between constituent models in a LOP, including manipulation of the training set and the feature set. We found the best method was based on the feature set, by creating constituent models from manually-specified intuitive partitions of the features. However, once these feature partitions were chosen, no additional diversity creating procedure was used. When the constituent models were trained, the parameters in each model were estimated independently, with no interaction with the parameters in other models. As we saw in Chapter 6, this approach worked well, but is it possible to encourage diversity in other ways, directly and automatically?

In this chapter we explore the answer to this question by investigating other ap-

proaches to creating diversity between the constituent models. We focus on the case of manipulating the training algorithm to directly encourage inter-model diversity. We call this approach **co-operative training**. In essence, because the training procedure encourages diversity between models, parameters in different models now become "aware" of each other and are trained interactively, rather than independently as in the approach of Chapter 6. This general avenue of exploration opens up a vast range of research possibilities, and we cannot investigate everything here. We therefore set ourselves the goal in this chapter of providing a *proof-of-concept* for the idea of using the CRF training algorithm to encourage diversity between constituent models in a LOP in order to improve the LOP's performance.

The rest of the chapter is structured as follows: In section 7.1 we give an overview of co-operative training, elaborating on the general idea introduced here. In section 7.2 we consider previous work that uses similar ideas but in other domains, appealing particularly to the arena of artificial neural networks (ANNs). Sections 7.3 and 7.4 describe the quantitative details behind co-operative training, while in section 7.5 we see how these details may be implemented in practise with a description of the software framework. Section 7.6 describes our experiments and the results we obtain, and, finally, in section 7.7 we summarise and conclude the chapter.

7.1 Co-operative Model Training

Figures 7.1 and 7.2 give a pictorial depiction of the idea behind co-operative training. In each figure the dark background represents the space of all models, while the target distribution is represented by q. This is the distribution that each model is attempting to represent. The light arrows between any two models in each figure represent an attractive influence, and hence a tendency for the models to move closer together in a KL-divergence sense. By contrast, the dark arrows represent a repulsive influence, and hence a tendency for the models to move further apart. The training algorithm involves optimising a criterion that encourages each model to be a good representation of q (light arrows forcing each constituent model to move towards q) while encouraging diversity between the models (dark arrows encouraging the constituent models to maintain their distance from each other). This corresponds in Equation 7.1 to reducing the E term while simultaneously increasing the A term. Figure 7.1 represents the situation early on during co-operative training. All constituent models, and therefore the LOP itself, are far from the target distribution. Then, as co-operative training

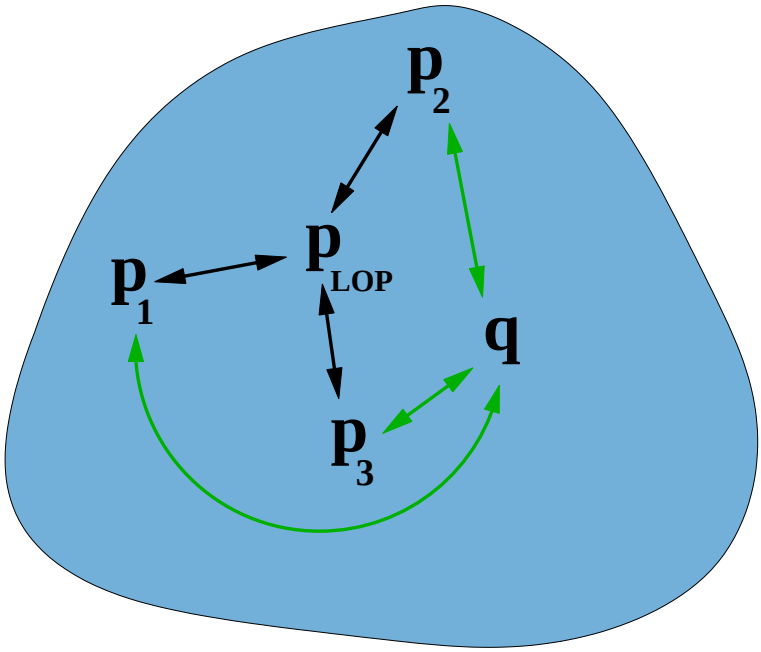

Figure 7.1: The LOP and its constituent models early in the co-operative training procedure.

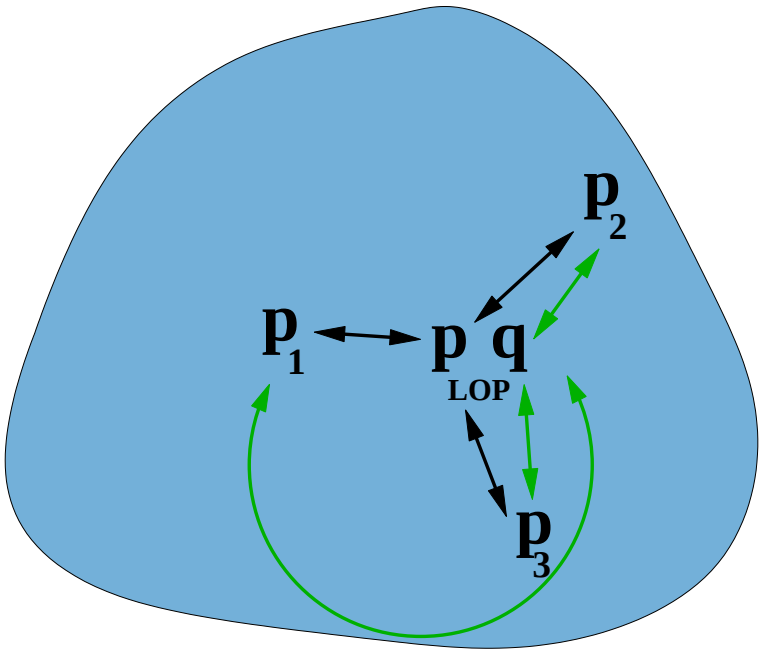

Figure 7.2: The LOP and its constituent models later in the co-operative training procedure.

progresses, we move closer to the situation depicted in Figure 7.2. Here each model has moved closer to the target distribution, forcing the LOP closer as well. Simultaneously, we have increased, or at least maintained, diversity between the constituent models using the repulsive influence represented by the dark arrows.

7.2 Related Work

The work most closely related to the ideas presented in this chapter comes from the field of neural networks. Within this domain, the general concept that diversity is important for creating ensembles of networks with low generalisation error has been well established for some years. One of the first formulations of this idea was presented by Krogh and Vedelsby (1995), who derived an **ambiguity decomposition** for neural network ensembles for regression. Since then numerous methods have been employed to try to introduce diversity into neural network ensembles. Many of these are analogous to the approaches we investigated for LOPs of CRFs in Chapter 6. Approaches have included using the feature set (Brown et al., 2002; Opitz, 1999; Zenobi and Cunningham, 2001), and manipulating the training data (Carney and Cunningham, 1999; Melville and Mooney, 2005). However, there has also been a smaller body of work that involves manipulating the training algorithm directly in order to encourage diversity. It is this work that is more specifically aligned with the ideas in this chapter. In the neural network community this training algorithm approach is generally known as **negative correlation learning** (Liu and Yao, 1999). An early example of the idea is provided by Rosen (1996), who adjusts the error function used by the back-propagation algorithm during network training to include a term containing correlations between pairs of networks. The presence of the term encourages networks to become decorrelated as training progresses. Liu and Yao (1999) use a similar idea by including what they call a **correlation penalty** term in the error function. This term encodes the correlation between the errors of each network and the network ensemble. The aim is to try to negatively correlate each network's error with the errors of the rest of the ensemble. Brown and Wyatt (2003) formalise the negative correlation learning idea by removing some of the assumptions made in the original presentation, and prove that certain bounds exist on the parameters that control the penalty term. The approaches of Liu and Yao (1999), and later Brown and Wyatt (2003), are therefore very similar to our approach in this chapter, where we use a **diversity penalty** to encourage diversity, or ambiguity, between the constituent CRF models in a LOP.

7.3 Training Algorithm

Design of the training algorithm for the co-operative training framework involves specifying an objective function and ascertaining how it may be optimised. In this section we construct a general form for the objective function and show how its derivatives may be derived.

7.3.1 Objective Function

Using Equation 7.1 as our guide, our aim is to formulate an objective function that simultaneously makes the E term decrease while forcing the A term to increase. We therefore create an objective comprising two parts:

1. For the first part we attempt to make the constituent models model the data well by encouraging the E term to be small. We use training data log-likelihood to do this. There are two clear candidates forms for this part. The first candidate involves a simple sum of the log-likelihoods for each constituent model, so has the form:

   $$\sum_{\alpha} LL_{\alpha}$$

   where LL_{α} denotes training data log-likelihood under model α.

   The second candidate involves a weighted sum of log-likelihoods for each constituent model. This therefore has the form:

   $$\sum_{\alpha} w_{\alpha} LL_{\alpha}$$

   where, as before, the w_{α} are the per-model weights. For the rest of the chapter we will consider the first candidate above as the default case, with the second candidate being a possible variant on this.

2. In the second part of the objective function we attempt to encourage diversity among the constituent models by constructing a term which explicitly penalises a small ambiguity. As in the first part, we experiment with two forms for this term. The first form, which we call the **L1 form**, looks like:

   $$-\frac{\gamma}{A}$$

where A is the ambiguity from Equation 7.1. The non-negative parameter γ controls the degree to which the penalty is effective. The second form, which we call the **L2 form**, looks like:

$$-\frac{\gamma}{A^2}$$

The terms **L1** and **L2** are coined from an analogy with conventional per-model priors. The term **L1 prior** is often used to signify a prior that is a linear in the model parameter in log-space, such as the Laplacian prior we saw in Chapter 5. Correspondingly, the term **L2 prior** is often used to signify a prior that is quadratic in the model parameters in log-space, such as a Gaussian prior. For the rest of the chapter we will consider the **L1** form of the penalty term as the default case, and the **L2** form as a possible variant. Note that both forms of the diversity penalty encourage the ambiguity to be as large as possible by penalising values close to zero. We could in addition penalise ambiguities that are too far away from some preferred value we have in mind for the ambiguity. This could be achieved by including additional penalty terms in the objective. We do not consider this level of refinement however.

The parameter γ is a little like a hyperparameter of a prior distribution (see prior families in Chapter 5). For large values of γ the effect of the prior begins to dominate in the objective function, putting more emphasis on diversity between models and less on individual model quality. Conversely, for small values of γ more emphasis is placed on model quality and diversity becomes less important. In the extreme case where γ is 0, the prior term is non-existent and the co-operative training framework collapses to the standard case where the individual models are trained independently, with no interaction between parameters in different models.

Putting together the two parts from above, the default form for our objective function becomes:

$$\Lambda(\lambda) = \sum_{\alpha} LL_{\alpha} - \frac{\gamma}{A} \tag{7.2}$$

7.3.2 Derivatives of the Objective Function

In order to be able to optimise the objective function described in the last section using the numerical optimisation routines we outlined in Chapter 3, we must be able to

evaluate the derivative of the objective with respect to the parameters we are adjusting. With co-operative training we are training model parameters in all constituent models together, simultaneously. Therefore we need an expression for the derivative of the objective function with respect to a particular parameter in a particular constituent model. Let us denote this by $\lambda_{\beta k}$: it is the kth parameter in model β.

To make the derivation clearer, let us denote the second term in our objective by a general function of the ambiguity and call it $f(A)$, like this:

$$\Lambda(\lambda) = \sum_{\alpha} LL_{\alpha} - f(A) \tag{7.3}$$

From our discussion in section 7.3.1 we know that $f(A)$ will be either $\frac{\gamma}{A}$ or $\frac{\gamma}{A^2}$. Therefore, we need to evaluate the following:

$$\begin{aligned} \frac{\partial \Lambda}{\partial \lambda_{\beta k}} &= \frac{\partial}{\partial \lambda_{\beta k}} \left[\sum_{\alpha} LL_{\alpha} - f(A) \right] \\ &= \sum_{\alpha} \frac{\partial LL_{\alpha}}{\partial \lambda_{\beta k}} - f'(A) \frac{\partial A}{\partial \lambda_{\beta k}} \end{aligned} \tag{7.4}$$

Clearly, the first term in Equation 7.4 is easy to evaluate because it is just the sum of the derivatives of the log-likelihoods of the data under each of the constituent models. In standard CRF training we need to calculate such a derivative for the single model we are using. Here we just need to do it across all the models, but the technology is the same. If we are using a weighted sum of log-likelihoods rather than an unweighted sum (the other possibility for the first term in the objective discussed above), the calculation is marginally more complex but does not cause any problem.

The second term in the derivative in Equation 7.4, however, is a little more complex. Clearly $f'(A)$ is either $-\frac{\gamma}{A^2}$ or $-\frac{2\gamma}{A^3}$, respectively, for the **L1** and **L2** forms of the diversity term in the objective. Provided we can evaluate A itself, neither of these two terms pose any problems. The potential problem, though, lies in the derivative of A, i.e. $\frac{\partial A}{\partial \lambda_{\beta k}}$.

Before attempting to evaluate $\frac{\partial A}{\partial \lambda_{\beta k}}$, let us first re-express A in a slightly different form to the one we have been using up to now. This re-representation will make the derivation a little easier. Currently, we have the following definition for A:

$$A = \sum_{\alpha} w_{\alpha} K(p_{\text{LOP}}, p_{\alpha}) \tag{7.5}$$

Expanding the KL-divergence term, we have:

$$A = \sum_{\alpha} w_\alpha \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p_{\text{LOP}}(\mathbf{s} \mid \mathbf{o}) \log \left[\frac{p_{\text{LOP}}(\mathbf{s} \mid \mathbf{o})}{p_\alpha(\mathbf{s} \mid \mathbf{o})} \right] \tag{7.6}$$

Moving the sum over the w_α through the expression, expanding the logarithm and dropping the arguments to p_{LOP} and p_α to save space, we have:

$$A = \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \sum_{\alpha} w_\alpha \left[p_{\text{LOP}} \log p_{\text{LOP}} - p_{\text{LOP}} \log p_\alpha \right] \tag{7.7}$$

Remembering that $\sum_\alpha w_\alpha = 1$, we get:

$$\begin{aligned} A &= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \left[p_{\text{LOP}} \log p_{\text{LOP}} - p_{\text{LOP}} \sum_{\alpha} w_\alpha \log p_\alpha \right] \\ &= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \left[p_{\text{LOP}} \log p_{\text{LOP}} - p_{\text{LOP}} \log \left[\prod_{\alpha} p_\alpha^{w_\alpha} \right] \right] \end{aligned} \tag{7.8}$$

But, as we defined earlier, $p_{\text{LOP}} = \frac{\prod_\alpha p_\alpha^{w_\alpha}}{Z_{\text{LOP}}}$, so we can re-express Equation 7.8 as:

$$\begin{aligned} A &= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \left[p_{\text{LOP}} \log p_{\text{LOP}} - p_{\text{LOP}} \log \left[p_{\text{LOP}} Z_{\text{LOP}} \right] \right] \\ &= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \left[p_{\text{LOP}} \log p_{\text{LOP}} - p_{\text{LOP}} \log p_{\text{LOP}} - p_{\text{LOP}} \log Z_{\text{LOP}} \right] \\ &= -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p_{\text{LOP}} \log Z_{\text{LOP}} \\ &= -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \log Z_{\text{LOP}} \end{aligned} \tag{7.9}$$

where, in last transition, we have used the fact that Z_{LOP} is not a function of the labelling $\mathbf{s}$ and of course $\sum_{\mathbf{s}} p_{\text{LOP}}(\mathbf{s} \mid \mathbf{o}) = 1$ for any $\mathbf{o}$. Remember that Z_{LOP} is a function of $\mathbf{o}$ so re-introducing the function notation, our new expression for A is:

$$A = -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \log Z_{\text{LOP}}(\mathbf{o}) \tag{7.10}$$

Now that we have an expression for A, we can take the derivative with respect to a general parameter $\lambda_{\beta k}$:

$$\frac{\partial A}{\partial \lambda_{\beta k}} = -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \frac{1}{Z_{\text{LOP}}(\mathbf{o})} \frac{\partial Z_{\text{LOP}}(\mathbf{o})}{\partial \lambda_{\beta k}} \tag{7.11}$$

Therefore we need to evaluate $\frac{\partial Z_{\text{LOP}}}{\partial \lambda_{\beta k}}$. Remember that Z_{LOP} is the normalising function in the definition of the LOP. Hence:

$$Z_{\text{LOP}}(\mathbf{o}) = \sum_{\mathbf{s}} \prod_{\alpha} p_{\alpha}(\mathbf{s} \mid \mathbf{o})^{w_{\alpha}} \tag{7.12}$$

So we need to take the derivative of $\prod_{\alpha} p_{\alpha}(\mathbf{s} \mid \mathbf{o})^{w_{\alpha}}$, but it is actually easier to take the derivative of the logarithm of this quantity. So we would like a way of expressing the derivative of a function of a variable in terms of the derivative of the logarithm of that function. For this we use a trick:

$$\frac{\partial}{\partial x} \log[f(x)] = \frac{1}{f(x)} \frac{\partial f}{\partial x} \tag{7.13}$$

and so:

$$\frac{\partial f}{\partial x} = f(x) \frac{\partial}{\partial x} \log[f(x)] \tag{7.14}$$

Taking the derivative of Z_{LOP} in Equation 7.11 using this trick we get:

$$\begin{aligned} \frac{\partial Z_{\text{LOP}}(\mathbf{o})}{\partial \lambda_{\beta k}} &= \sum_{\mathbf{s}} \left(\prod_{\alpha} p_{\alpha}^{w_{\alpha}} \right) \frac{\partial}{\partial \lambda_{\beta k}} \left[\sum_{\alpha} w_{\alpha} \log p_{\alpha} \right] \\ &= \sum_{\mathbf{s}} \left(\prod_{\alpha} p_{\alpha}^{w_{\alpha}} \right) \sum_{\alpha} w_{\alpha} \delta_{\alpha\beta} \frac{\partial \log p_{\alpha}}{\partial \lambda_{\beta k}} \\ &= \sum_{\mathbf{s}} \left(\prod_{\alpha} p_{\alpha}^{w_{\alpha}} \right) w_{\beta} \frac{\partial \log p_{\beta}}{\partial \lambda_{\beta k}} \end{aligned} \tag{7.15}$$

Now, $p_{\beta}(\mathbf{s} \mid \mathbf{o})$ is the probability density function for a standard CRF, so $\log p_{\beta}$ has the form:

$$\begin{aligned} \log p_{\beta}(\mathbf{s} \mid \mathbf{o}) &= \log \left[\frac{1}{Z_{\beta}(\mathbf{o})} \exp \left(\sum_{n} \sum_{j} \lambda_{\beta j} f_{\beta j}(\mathbf{s}, \mathbf{o}, n) \right) \right] \\ &= \sum_{n} \sum_{j} \lambda_{\beta j} f_{\beta j}(\mathbf{s}, \mathbf{o}, n) - \log Z_{\beta}(\mathbf{o}) \end{aligned} \tag{7.16}$$

where Z_{β} is the normalisation constant for model β and $f_{\beta j}(\mathbf{s}, \mathbf{o}, n)$ refers to the jth feature in model β on clique n in a sequence with observation $\mathbf{o}$ and labelling $\mathbf{s}$. So, taking the derivative with respect to $\lambda_{\beta k}$ (which we need in Equation 7.15 above), we get:

$$\begin{aligned}\frac{\partial \log p_\beta}{\partial \lambda_{\beta k}} &= \frac{\partial}{\partial \lambda_{\beta k}}\left[\sum_n \sum_j \lambda_{\beta j} f_{\beta j}(\mathbf{s},\mathbf{o},n) - \log Z_\beta(\mathbf{o})\right] \\ &= \sum_n f_{\beta k}(\mathbf{s},\mathbf{o},n) - \frac{1}{Z_\beta(\mathbf{o})}\frac{\partial Z_\beta(\mathbf{o})}{\partial \lambda_{\beta k}} \end{aligned} \tag{7.17}$$

The second term on the right-hand side here can be evaluated relatively easily, as follows:

$$\begin{aligned}\frac{1}{Z_\beta(\mathbf{o})}\frac{\partial Z_\beta(\mathbf{o})}{\partial \lambda_{\beta k}} &= \frac{1}{Z_\beta(\mathbf{o})}\frac{\partial}{\partial \lambda_{\beta k}}\left[\sum_{\mathbf{s}'} \exp\left(\sum_{n'}\sum_{j'} \lambda_{\beta j'} f_{\beta j'}(\mathbf{s}',\mathbf{o},n')\right)\right] \\ &= \frac{1}{Z_\beta(\mathbf{o})}\sum_{\mathbf{s}'}\left[\sum_{n'} f_{\beta k}(\mathbf{s}',\mathbf{o},n')\right]\exp\left(\sum_{n'}\sum_{j'} \lambda_{\beta j'} f_{\beta j'}(\mathbf{s}',\mathbf{o},n')\right) \\ &= \sum_{\mathbf{s}'}\left[\sum_{n'} f_{\beta k}(\mathbf{s}',\mathbf{o},n')\right]\frac{1}{Z_\beta(\mathbf{o})}\exp\left(\sum_{n'}\sum_{j'} \lambda_{\beta j'} f_{\beta j'}(\mathbf{s}',\mathbf{o},n')\right) \\ &= \sum_{\mathbf{s}'} p_\beta(\mathbf{s}'\,|\,\mathbf{o})\left[\sum_{n'} f_{\beta k}(\mathbf{s}',\mathbf{o},n')\right] \end{aligned} \tag{7.18}$$

Therefore, our expression for $\frac{\partial \log p_\beta}{\partial \lambda_{\beta k}}$ in Equation 7.17 becomes:

$$\frac{\partial \log p_\beta}{\partial \lambda_{\beta k}} = \sum_n f_{\beta k}(\mathbf{s},\mathbf{o},n) - \sum_{\mathbf{s}'} p_\beta(\mathbf{s}'\,|\,\mathbf{o})\left[\sum_{n'} f_{\beta k}(\mathbf{s}',\mathbf{o},n')\right] \tag{7.19}$$

and, in turn, $\frac{\partial Z_{\text{LOP}}(\mathbf{o})}{\partial \lambda_{\beta k}}$ in Equation 7.15 becomes:

$$\frac{\partial Z_{\text{LOP}}(\mathbf{o})}{\partial \lambda_{\beta k}} = \sum_{\mathbf{s}}\left(\prod_\alpha p_\alpha^{w_\alpha}\right) w_\beta\left[\sum_n f_{\beta k} - \sum_{\mathbf{s}'} p_\beta(\mathbf{s}'\,|\,\mathbf{o})\left[\sum_{n'} f_{\beta k}\right]\right] \tag{7.20}$$

Expanding the outer bracket in Equation 7.20 above and substituting into our original expression for $\frac{\partial A}{\partial \lambda_{\beta k}}$ in Equation 7.11, we finally arrive at:

$$\begin{aligned}\frac{\partial A}{\partial \lambda_{\beta k}} &= -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}}\left(\frac{\prod_\alpha p_\alpha^{w_\alpha}}{Z_{\text{LOP}}(\mathbf{o})}\right) w_\beta \sum_n f_{\beta k}(\mathbf{s},\mathbf{o},n) \\ &\quad + \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}}\left(\frac{\prod_\alpha p_\alpha^{w_\alpha}}{Z_{\text{LOP}}(\mathbf{o})}\right) w_\beta \sum_{\mathbf{s}'} p_\beta(\mathbf{s}'\,|\,\mathbf{o})\left[\sum_{n'} f_{\beta k}(\mathbf{s}',\mathbf{o},n')\right] \\ &= -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p_{\text{LOP}}(\mathbf{s}\,|\,\mathbf{o})\, w_\beta \sum_n f_{\beta k}(\mathbf{s},\mathbf{o},n) \\ &\quad + \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p_{\text{LOP}}(\mathbf{s}\,|\,\mathbf{o})\, w_\beta \sum_{\mathbf{s}'} p_\beta(\mathbf{s}'\,|\,\mathbf{o})\left[\sum_{n'} f_{\beta k}(\mathbf{s}',\mathbf{o},n')\right] \end{aligned} \tag{7.21}$$

In the second term on the right-hand side here, we are taking a weighted sum $\sum_{\mathbf{s}}$ over labellings $\mathbf{s}$ weighted by the $p_{\text{LOP}}(\mathbf{s} \mid \mathbf{o})$ distribution. However, as the argument to the sum (i.e. the w_β and inner $\sum_{\mathbf{s}'}$ sum) is not a function of the labelling $\mathbf{s}$, it can be taken outside the $\sum_{\mathbf{s}}$ sum and the sum itself just collapses to 1. Hence we can simplify the whole expression to:

$$\begin{aligned}
\frac{\partial A}{\partial \lambda_{\beta k}} &= -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p_{\text{LOP}}(\mathbf{s} \mid \mathbf{o}) w_\beta \sum_n f_{\beta k}(\mathbf{s}, \mathbf{o}, n) \\
&\quad + \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) w_\beta \sum_{\mathbf{s}'} p_\beta(\mathbf{s}' \mid \mathbf{o}) \sum_{n'} f_{\beta k}(\mathbf{s}', \mathbf{o}, n') \\
&= -w_\beta \mathrm{E}_{p_{\text{LOP}}}\left[f_{\beta k}\right] + w_\beta \mathrm{E}_{p_\beta}\left[f_{\beta k}\right] \\
&= -w_\beta \left[\mathrm{E}_{p_{\text{LOP}}}\left[f_{\beta k}\right] - \mathrm{E}_{p_\beta}\left[f_{\beta k}\right]\right] \qquad (7.22)
\end{aligned}$$

Therefore the derivative of A with respect to a general kth parameter $\lambda_{\beta k}$ in one of the constituent models β is actually the difference between the expected count of the associated feature under that model and the expected count of the associated feature under the LOP. Armed with this we can now evaluate the derivative of the entire objective function in (7.4). It becomes:

$$\frac{\partial \Lambda}{\partial \lambda_{\beta k}} = \sum_\alpha \frac{\partial LL_\alpha}{\partial \lambda_{\beta k}} + f'(A) w_\beta \left[\mathrm{E}_{p_{\text{LOP}}}\left[f_{\beta k}\right] - \mathrm{E}_{p_\beta}\left[f_{\beta k}\right]\right] \qquad (7.23)$$

We evaluate this expression on each iteration of co-operative training, and pass it to the number optimisation routines along with the value of the objective itself.

From the form of the objective function and its derivatives derived above, we suspect that the objective is convex in the model parameters. However, so far we have not been able to prove this.

7.4 Decoding

The specification of an alternative objective function and evaluation of its derivatives make co-operative training a little more complex than the basic case of LOP training we introduced in Chapter 6. However, once the constituent models have been trained within the co-operative training framework, decoding the models under a LOP is identical to before.

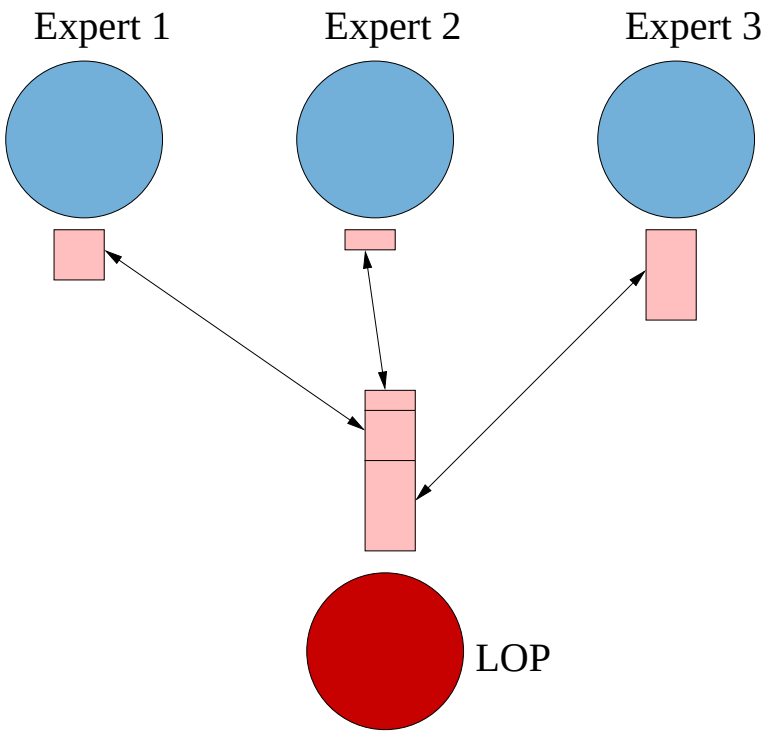

Figure 7.3: Schematic representation of architecture for co-operative training.

7.5 Implementation

In this section we give both a schematic description of the architecture, and look briefly at the specifics of the software implementation.

7.5.1 Schematic Representation

7.5.1.1 Training

Figure 7.3 gives a schematic representation of the architecture we use for co-operative training. For the purposes of illustration in the diagram, we see a LOP with three constituent models, or experts. In addition to the constituent models, we think of the LOP itself as having a distinct identity, with its own node. During training (and decoding) information flows back and forth between each constituent model and the LOP. This information can be single valued data, like a double representing a per-model weight, or a vector of values, like a vector of doubles representing a parameter vector or gradient vector. The illustration depicts one such example communication. It shows the subsets of a vector residing at the LOP node being scattered to different constituent model nodes. Each subset of the vector contains information relevant to

the respective constituent model.

During training we need to pass the value of the objective function from Equation 7.3 and its gradient vector from Equation 7.23 to the libraries containing the numerical optimisation routines. This only need occur at one node, and we choose the LOP node as the most natural choice for this. Therefore the LOP node must be aware, and be able to communicate, all terms in Equations 7.3 and 7.23 to the libraries. In order for this to occur, a sequence of communication operations between the LOP node and the constituent model nodes takes place during each iteration of the co-operative training. The low-level details are a little intricate, but in essence the following takes place in each iteration:

1. An updated **global parameter vector** containing all parameters in all constituent models, and the per-model weights, is received by the LOP node from the libraries.

2. Subsets of this parameter vector, each containing model parameters and a per-model weight, are sent to the respective constituent model nodes.

3. Each constituent model α calculates its **distribution vector**, i.e. a vector representing p_α. This encodes probabilities for all labellings of sequences in the training data given the updated values (for the current iteration) of the model parameters.

4. Each constituent model α communicates its distribution vector to the LOP node.

5. The constituent model nodes and the LOP node process their data in parallel:

   - Each constituent model α calculates LL_α, $\frac{\partial LL_\alpha}{\partial \lambda_{\beta k}}$ and the vector representing $\mathrm{E}_{p_\alpha}\left[f_{\beta k}\right]$.
   - The LOP node calculates its **distribution vector** representing p_{LOP}, the ambiguity A, $f(A)$ and the vector representing $\mathrm{E}_{p_{\mathrm{LOP}}}\left[f_{\beta k}\right]$.

6. Each constituent model α sends LL_α, $\frac{\partial LL_\alpha}{\partial \lambda_{\beta k}}$ and the vector representing $\mathrm{E}_{p_\alpha}\left[f_{\beta k}\right]$ to the LOP node.

7. The LOP node gathers the data together and evaluates the objective function from Equation 7.3 and its gradient vector from Equation 7.23. It then sends these to the the optimisation libraries, and the process starts over.

7.5.1.2 Decoding

As mentioned in section 7.4 above, the decoding of a LOP consisting of constituent models with co-operatively trained parameters is no more complex than the LOP decoding described in Chapter 6. However, given the architecture required for the co-operative training algorithm, it is easy to decode (both schematically and in practice) within the same framework. The decoding therefore involves the following steps:

1. Each constituent model α reads in a parameter vector from the training stage (the final parameter vector or one representing a particular iteration) and also reads in a specified dataset for decoding. Using the parameter vector, each model calculates its **distribution vector**, i.e. the vector representing p_α for that dataset.

2. Each constituent model α communicates its distribution vector to the LOP node.

3. The LOP node gathers together the constituent model distribution vectors and weights them, so forming and the **distribution vector** representing p_{LOP}.

4. The LOP node undertakes the decoding of the specified dataset. The actual decoding step itself therefore occurs only at the LOP node.

7.5.2 Software Implementation

The software we have written for co-operative training is implemented in C++ and Perl. Perl scripts generally control high-level behaviour. For example, a single Perl script would define all the settings and necessary actions to be taken for particular experiment. The C++ code by contrast controls the lower-level, computationally intensive behaviour for the training and decoding of the models. The C++ code is divided into two binaries, one for training and one for decoding. Each of these is evoked from a Perl script with the correct flag settings for the experiment at hand.

The C++ code for model training and decoding is parallelised, and is designed to work on a cluster of machines. Use of multiple machines is a necessity because for co-operative training to be efficient, a representation of each constituent model must be maintained in memory during the training process (as can be seen from our calculations of the objective function and its gradient vector earlier). In general this memory requirement is far too large for any single machine that we have access to. We make use of an implementation of the **message passing interface** (MPI)[1] protocols to

[1] http://www-unix.mcs.anl.gov/mpi

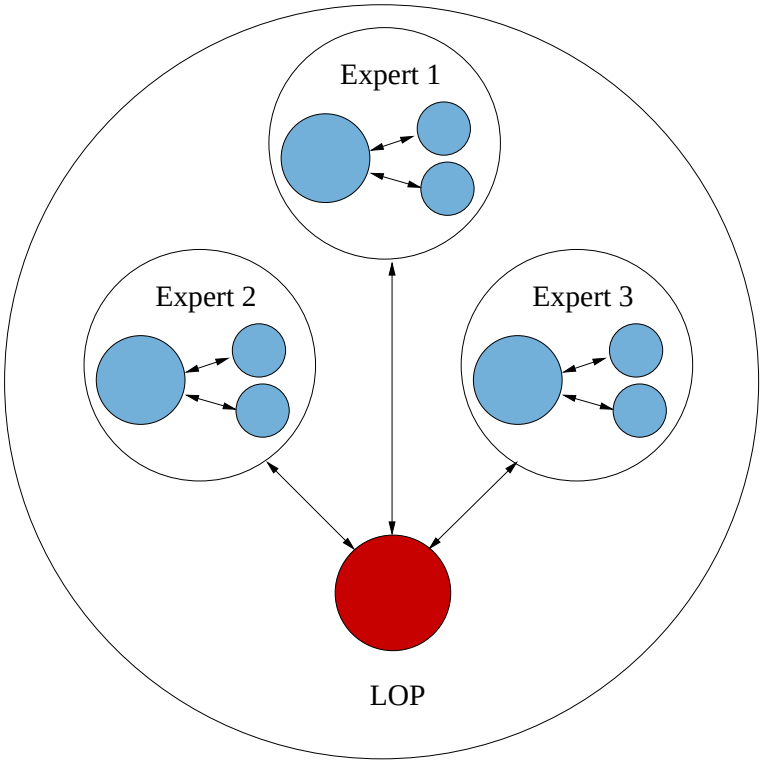

Figure 7.4: Software architecture for co-operative training.

aid communication between the nodes in the cluster. The MPI libraries are described in more detail in Chapter 3.

Figure 7.4 gives a more software-specific representation of the architecture than the more general view in Figure 7.3. In the simplest case, each constituent model occupies only one node of the cluster, with the LOP node also occupying a node. However, for some tasks and/or datasets we may need to parallelise each constituent model across multiple nodes. This situation is illustrated in Figure 7.4, with each constituent model being spread across three nodes. When this extra level of parallelisation is in effect, the LOP node communicates with one "representative" node from each constituent model. We call this representative node the **gateway node**. In Figure 7.4, the gateway nodes are represented by the larger circles in each constituent model grouping. Therefore, there are really two levels of communication: communication within each constituent model, between the constituent model's gateway node and its other nodes, and communication at the level of the LOP, between the gateway nodes and the LOP node itself. The MPI infrastructure makes this two-level communication easy to implement in an elegant way. We do this by defining an MPI communicator for each constituent model, including only the nodes in that constituent model. These communicators are

represented by the inner circles in Figure 7.4. We then define one "global" communicator that contains only the gateway nodes and the LOP node. This communicator is represented by the outer circle in Figure 7.4. Defining these communicators means we can use a C++ class structure that wraps the MPI communicator structure and allows a single set of C++ class member functions to implement different MPI behaviour polymorphically depending on the level of abstraction of the message passing at a particular time.

7.6 Experiments

In our experiments our primary goal is to establish the effectiveness of the diversity penalty term in creating better performing LOPs. Our secondary goal is to conduct a preliminary investigation into the sensitivity of the training process to various changes in the setup, such as the use of a different log-likelihood term in the objective, different forms for the penalty term, and use of different expert sets. In order to handle these possibilities we define a **default configuration** and then investigate variants of this default by modifying a single aspect in each case. All of the experiments are conducted on the NER task described in Chapter 3.

7.6.1 Default Configuration

The default configuration for our experiments consists of the following:

1. Default objective function defined earlier, i.e. $\sum_\alpha LL_\alpha - \frac{\gamma}{A}$
2. Unregularised constituent models
3. Uniform per-model weights
4. STANDARD and SIMPLE constituent models only

We could start the co-operative training from the zero parameter vector for each constituent model in the LOP. However, this would make the process unnecessarily inefficient because we already have a set of good candidate vectors for starting the training from the experiments in previous chapters. Therefore, we take advantage of this set by seeding each constituent model with a **priming vector**. This vector, for a given constituent model, is the one that achieved the best F score on the development set

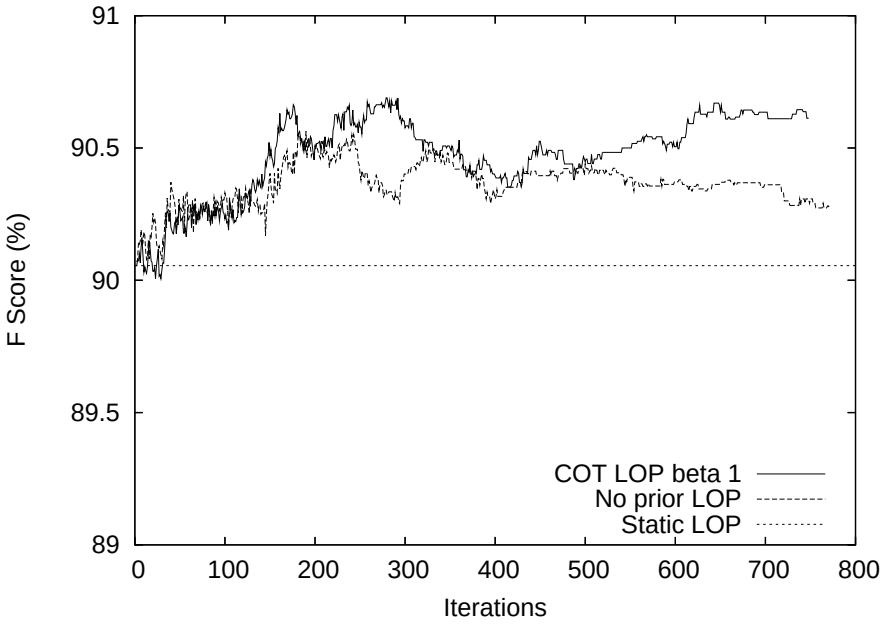

Figure 7.5: Development set F scores for an unregularised co-operatively trained LOP.

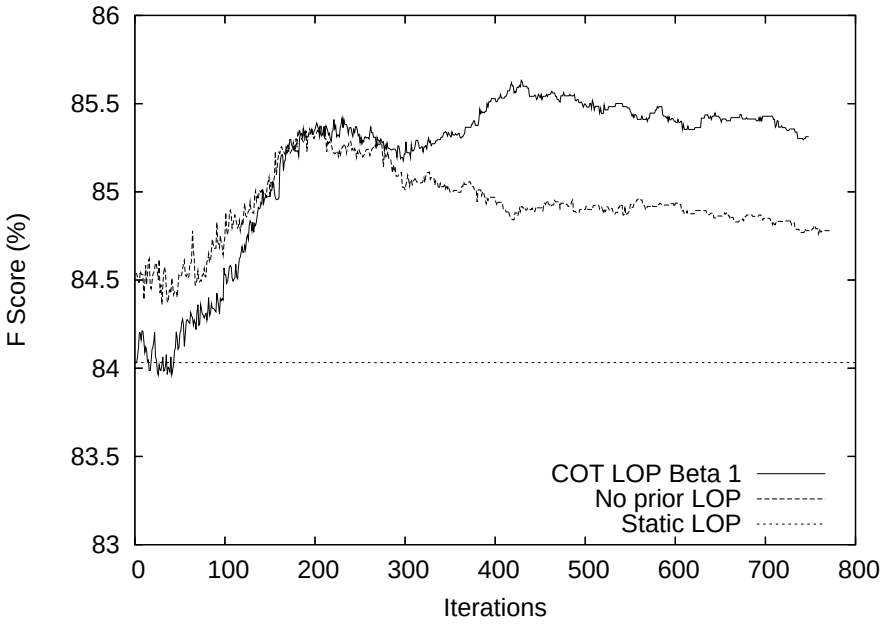

Figure 7.6: Test set F scores for an unregularised co-operatively trained LOP.

when the model was trained originally (independently of any other model). We use this seeding technique in this section, and the other sections that follow.

Figure 7.5 shows development set F scores for the default configuration as the co-operative training progresses. It is represented by the highest of the three lines in the figure. The lowest line, the straight one, shows the F score for a single LOP created using the priming vectors for each constituent model and using uniform per-model weights. Hence this is one of the LOPs we first reported on in section 6.5.2.2. The middle line is intended to act as a baseline for the performance of the default configuration. This line shows what happens when the constituent models continue to be trained, unregularised, starting from their priming vectors but independently of each other, with no interaction. The F scores that the corresponding LOP achieves are higher than the score it achieved at the priming vector point (i.e. represented by the straight line). However, this is just a facet of the optimisation routine. Having started to re-train each model at its priming vector, the training procedure takes the models into a region of the parameter space where the resulting LOP happens to do better than it did previously when the constituent models were each trained from the zero parameter vector. This is not an important point though. The important point is that the default configuration, trained using a diversity penalty, outperforms this baseline. Figure 7.6 shows the test set F scores corresponding to the development set scores in Figure 7.5. We can see that, once again, the default configuration score is significantly higher than that of the baseline. We therefore have evidence that the presence of the diversity penalty *is* achieving the intended effect of making the constituent models more diverse and so improving the corresponding LOP.

The default configuration shown in the figures corresponds to a γ value of 1, which we found using manual tuning on the development set. However, further improvements are likely to be possible with a more thorough search of the γ-space. We use the same manual tuning approach in each of the following sections.

7.6.2 Using Regularised Experts

In Chapter 6 we looked at LOPs created from regularised constituent models, and compared them to LOPs comprising only unregularised models. We found that the unregularised LOPs performed better. We now make the same comparison with co-operatively trained constituent models. To do this we undertake co-operative training with a diversity penalty, but simultaneously regularise the constituent models using separate Gaussian priors. For each model we use the (parameter-independent) optimal Gaussian variance, found using the development set when the models were trained in-

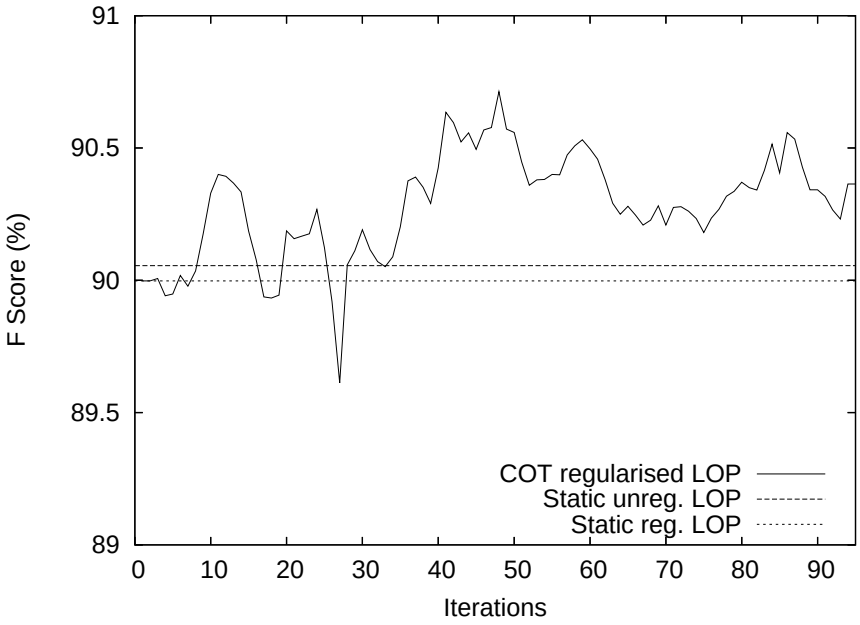

Figure 7.7: Development set F scores for a regularised co-operatively trained LOP.

dependently. For the STANDARD model this value is 44, while for the SIMPLE model it is 100. Of course, these values are not necessarily optimal for the co-operatively trained constituent models as they were established when each model was trained independently, but they should form good approximations to the optimal values.

Figure 7.7 shows development set F scores for the default configuration modified to use regularised constituent models. The figure also shows two non-co-operatively trained LOPs, one with unregularised constituent models (as seen in Figure 7.5) and the other with regularised constituent models, combined with uniform per-model weights. These are LOPs we presented results for in Chapter 6. The general performance profile of the co-operatively trained LOP is similar to the one observed in Figure 7.5 with the unregularised constituent model LOP. We therefore have additional evidence that the diversity penalty improves the LOP, this time in the presence of alternative, per-model regularisation. As we observed in Chapter 6, however, the LOP with regularised constituent models underperforms the LOP with unregularised models. Once again, we attribute this to the subtle trade-off between model quality and LOP diversity, the regularised models being better models of the underlying distribution but also being less diverse because they are closer to the uniform distribution.

7.6.3 Using Weighted Log-Likelihoods

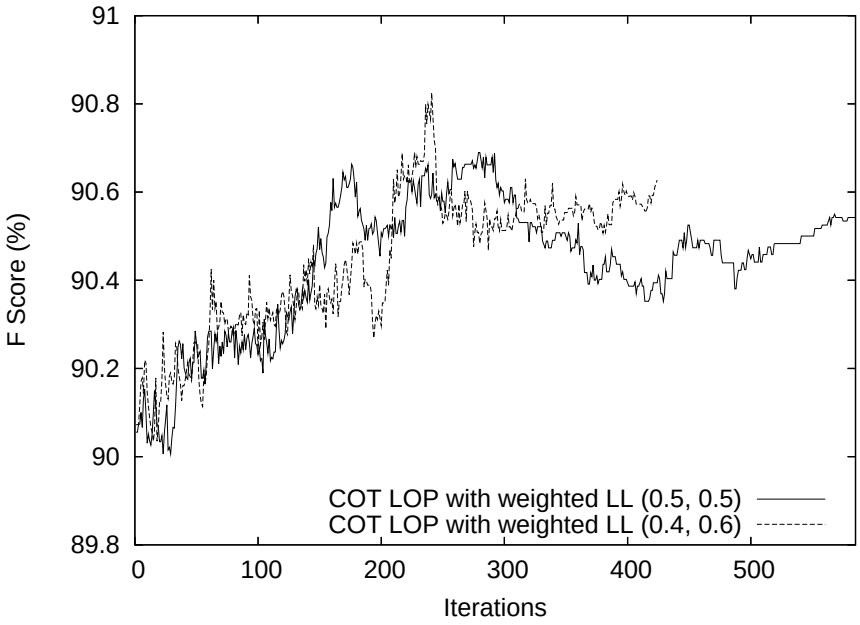

Figure 7.8: Development set F scores for an unregularised co-operatively trained LOP using a weighted sum of log-likelihoods in the objective function.

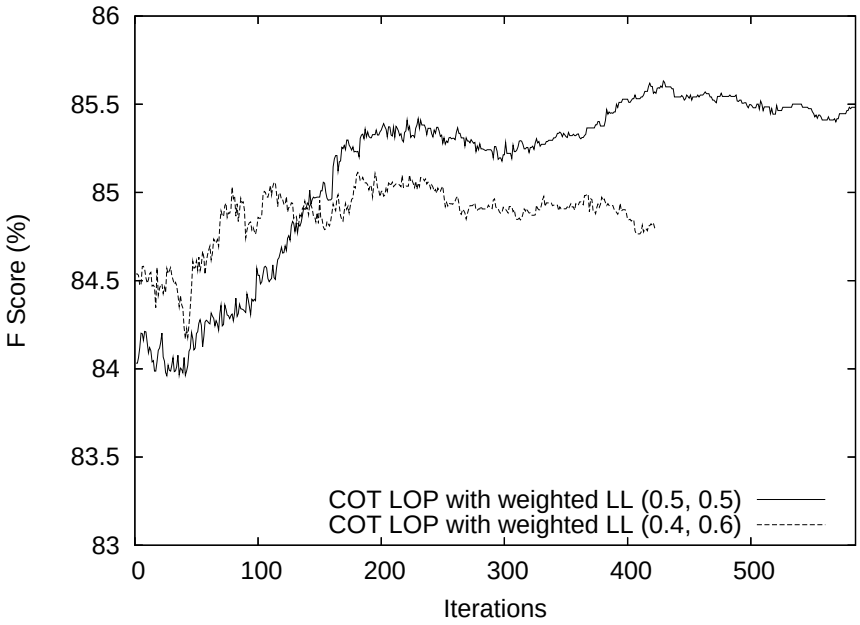

Figure 7.9: Test set F scores for an unregularised co-operatively trained LOP using a weighted sum of log-likelihoods in the objective function.

In the default configuration the first term in the objective function is the sum of the log-likelihoods of the constituent models. In this section we investigate the effect of replacing that term with a *weighted* sum of log-likelihoods, $\sum_\alpha w_\alpha LL_\alpha$. Note that the weights used in this weighted sum are the same as the weights used in the calculation of the LOP distribution during training and decoding, so we are also using non-uniform weights in the LOP generally, unlike in the default configuration.

Figure 7.8 shows development set F scores for LOPs co-operatively trained with a non-uniformly weighted sum of log-likelihoods. Figure 7.9 shows the corresponding test set F scores. Clearly, the lines labelled (0.5, 0.5) just represent the default configuration. From the figures we can see that the choice of weights can make a significance difference to the performance of the resulting LOP when co-operatively trained. For example, the default configuration performs significantly better on the test set than that the LOP with weights (0.4, 0.6), although on the development the difference is less clear. Other weight combinations produce similarly varied performance profiles. This suggests that it would be worthwhile to devise a strategy for automatically finding optimal weight choices. Typically such a procedure would be run as a separate process, offline, before the co-operative training commences. However, this extension is beyond the scope of the thesis, and we leave it for future work.

7.6.4 Using Other Diversity Penalties

In the co-operative training framework, the second term in the objective function is the diversity penalty. When we described the diversity penalty earlier in the chapter we proposed two forms that it might take, although many other choices are possible. The default configuration contains the **L1** form of the penalty term, but we also suggested an **L2** form. In this section we look at the effect of using this **L2** form.

Figure 7.10 shows development set F scores for LOPs co-operatively trained with both **L1** and **L2** forms for the diversity penalty. Figure 7.11 shows the corresponding test set F scores. We can see from the figures that neither form consistently outperforms the other across both datasets. The **L1** form tends to do better than the **L2** on the development set while the situation is reversed on the test set. This suggests that alternative forms for the diversity penalty, such as the **L2** form, can also be effective in improving the LOP with co-operative training. However, much additional work is required in this area to ascertain the precise properties of an effective penalty. The functional form of the penalty effects its influence in the objective function. Too strong

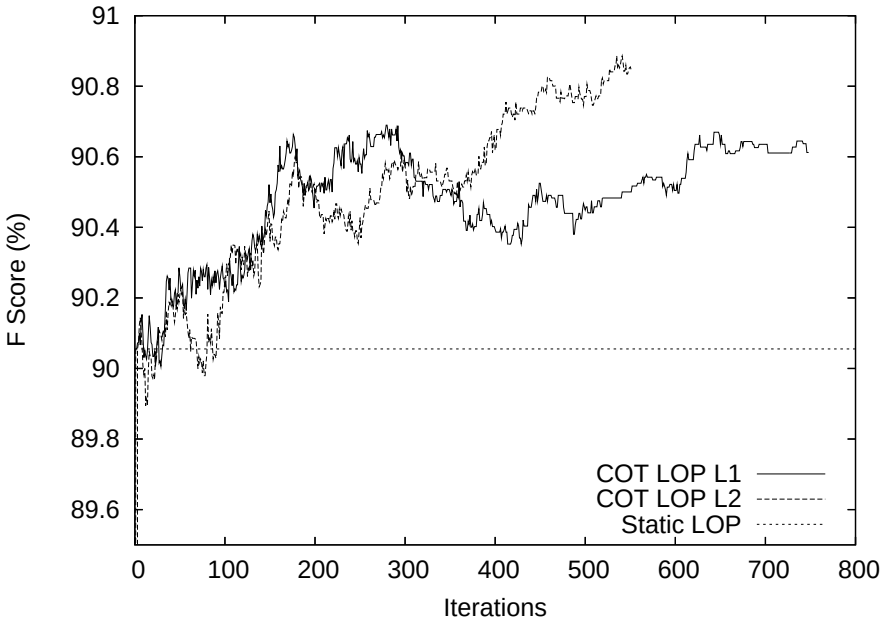

Figure 7.10: Development set F scores for an unregularised co-operatively trained LOP using an L2 diversity penalty in the objective function.

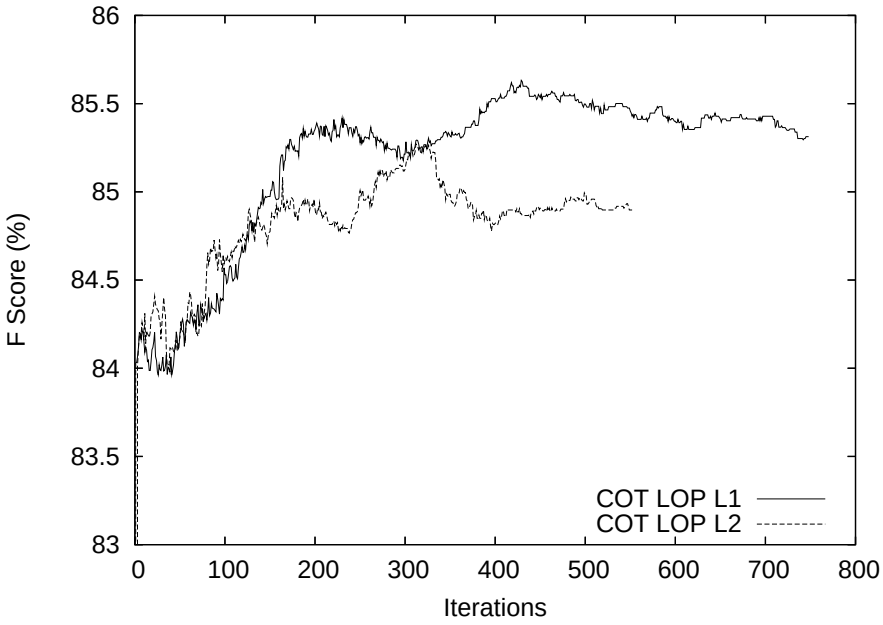

Figure 7.11: Test set F scores for an unregularised co-operatively trained LOP using an L2 diversity penalty in the objective function.

a penalty can lead to ambiguity dominating over accuracy, whereas too weak a penalty can have the opposite effect. Although the general influence of the penalty may be

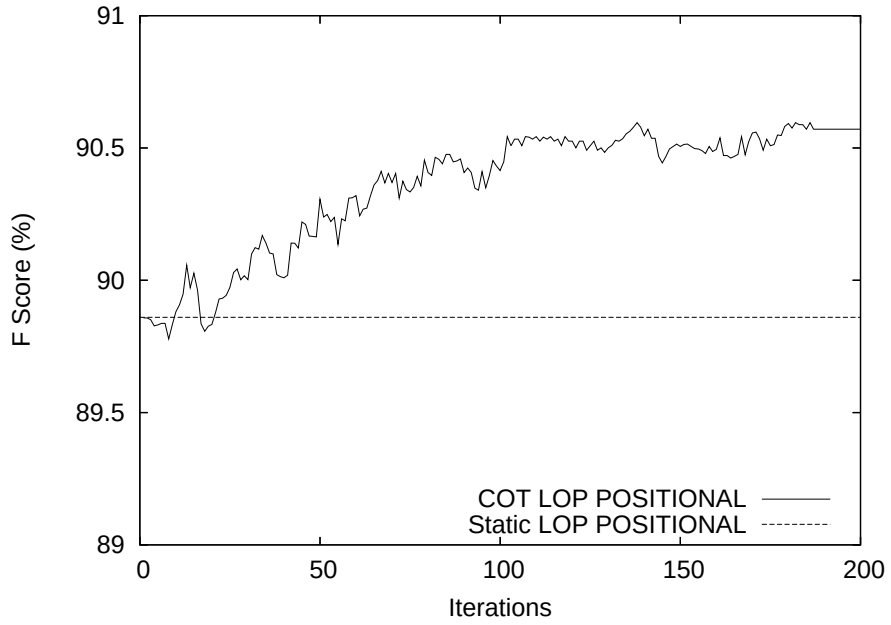

Figure 7.12: Development set F scores for an unregularised co-operatively trained LOP using POSITIONAL experts.

controlled through a parameter such as the γ we discussed earlier, this form of control may have too crude a calibration if the functional form of the penalty is not suitable.

7.6.5 Other Expert Sets

All the co-operative training results we have presented so far in this chapter have involved LOPs comprising a single expert set consisting of the STANDARD and SIMPLE models. In this section we investigate whether the same positive results for the diversity penalty extend to other expert sets. In particular we look at the LABEL and POSITIONAL expert sets that we described in Chapter 6.

Figure 7.12 shows development set F scores for LOPs co-operatively trained with POSITIONAL experts. Figure 7.13 shows the test set F scores. Figures 7.14 and 7.15 give the corresponding graphs for the LABEL expert set. The straight line in each graph represents the score the corresponding LOP obtained when the constituent models were trained independently. From the four graphs we can see that we obtain similar behaviour with the POSITIONAL and LABEL expert sets, when co-operatively trained with a diversity penalty, as we do with the SIMPLE expert set used earlier. We therefore have additional evidence that the diversity penalty is achieving its purpose.

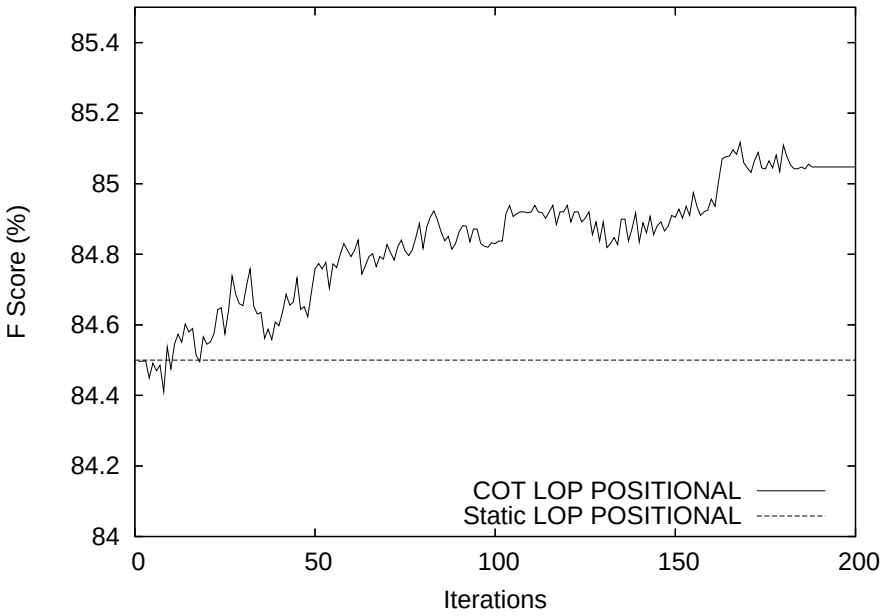

Figure 7.13: Test set F scores for an unregularised co-operatively trained LOP using POSITIONAL experts.

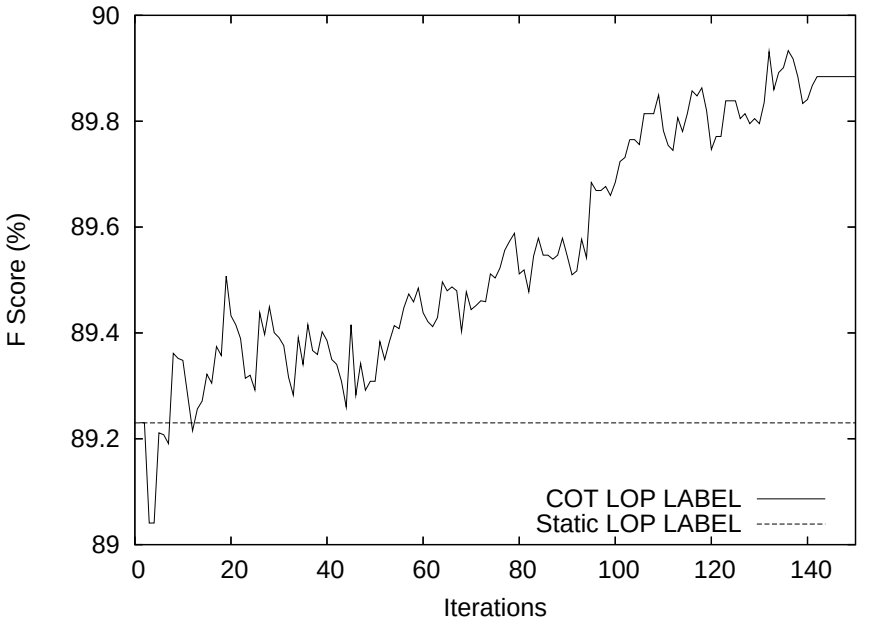

Figure 7.14: Development set F scores for an unregularised co-operatively trained LOP using LABEL experts.

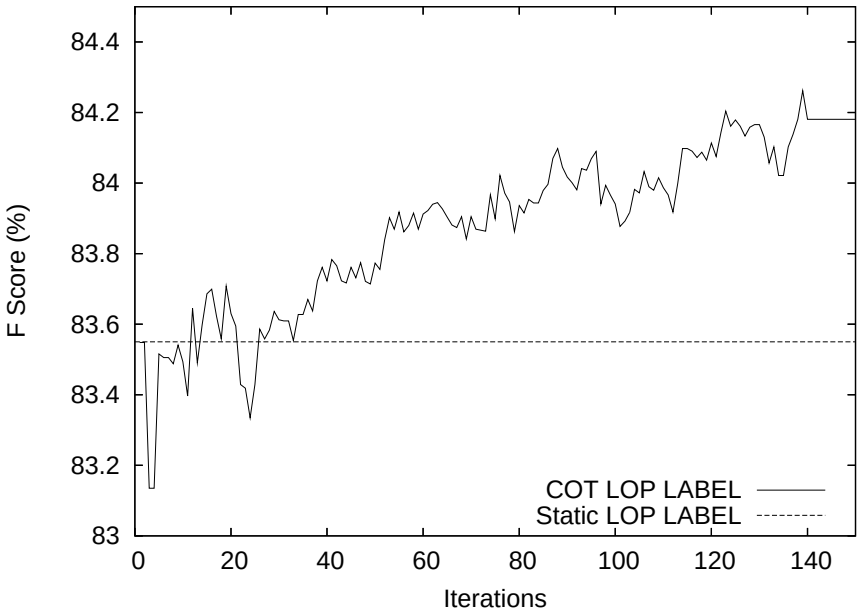

Figure 7.15: Test set F scores for an unregularised co-operatively trained LOP using LABEL experts.

7.6.6 Joint Parameter and Weight Training

In this chapter we have focused only on co-operative training of the constituent models, and have ignored the task of additionally learning the per-model weights in the LOP (the w_α). It is possible, however, to extend the co-operative training framework to include weight training. This would involve evaluation of the derivatives of the objective function with respect to the model weights. Theoretical calculation of these derivatives is actually simpler than the parameter derivative derivation we presented in section 7.3.2, and subsequent modification to the framework would not be huge. The work is, however, beyond the scope of the thesis, and so we leave it to future work.

7.7 Summary

In this chapter we have explored how to encourage diversity between constituent models in a LOP by manipulating the CRF training algorithm directly. In particular, we introduced the idea of co-operative training of LOP constituent models, where parameters in different models interact during the training process. We encourage diversity between the models using a specially formulated objective function which includes a diversity penalty term.

Our main results show that the presence of a diversity term in the objective can indeed lead to LOPs with improved performance. We have shown this to be the case for LOPs created from a number of the expert sets we introduced in Chapter 6. The work in this chapter therefore represents a *proof-of-concept* for the idea of manipulating diversity during training. The space of possibilities for more thorough investigation of the ideas we have presented in this chapter, and extensions to them, is vast. We have really only scratched the surface here, and many aspects warrant more thorough investigation. For example, our results show that manipulating the log-likelihood part of the objective function can affect the LOP's performance to a significant degree. We were using a weighted combination of constituent model log-likelihoods, but other functions of the log-likelihood are possible. We proposed two candidate forms for the diversity penalty, but many other possibilities exist. In addition, we only looked at co-operatively training the parameters of constituent models in this chapter, but it is also possible to incorporate the search for optimal per-model weights into the framework. We leave all these issues as possible avenues for future research.

Chapter 8

Applications of LOPs: Highly Discriminative Features

In Chapter 6 we introduced the logarithmic opinion pool for CRFs and showed how it may be seen as an alternative to conventional regularisation of a CRF with a prior. Our explanation for the good performance of the LOP was based on the **ambiguity decomposition** in Equation 6.13. The decomposition motivated the idea that a good LOP contains constituent models that are both individually accurate and mutually diverse in a KL-Divergence sense.

In this chapter we change angle slightly and view the LOP from an alternative standpoint. In particular, a LOP may be seen as a solution to an important but generally under-appreciated problem that occurs when training CRFs, and log-linear models in general, with highly discriminative features. A highly discriminative feature in this context is a feature which is often sparse, and whose observation context is highly correlated with the corresponding label that it encodes. Hence the observation context is a strong predictor of the label. An example of such a feature could be a gazetteer feature, and indeed, as we shall see later in the chapter, in our experiments we use gazetteer features as illustrative examples.

We demonstrate in the following sections that although inclusion of such highly discriminative features in a model in general leads to improved performance, it can also cause negative effects. These effects stem from an over-reliance by the model on these features to explain the data. By including the new features in the model, we transfer explanatory power to these features away from the existing features, so making the existing features less expressive. This can lead to certain types of labelling errors during decoding. Our conclusion is that to fully harness the power of highly

discriminative features a more careful training regime must be used, and we show that training and then decoding using a LOP is a solution for this. At the end of the chapter we show that in addition to gazetteer features, similar behaviour may be obtained with other subsets of discriminative features, and we show how such subsets may be identified.

8.1 Gazetteers and Gazetteer Features

In order to demonstrate the problems with discriminative features we have outlined above, we need to be able to identify a subset of features within a model, or a set of features that could be added to a model, that have this discriminative property. To do this we make use of **gazetteer features** as a running example throughout the chapter. In this section we introduce the idea behind gazetteers, and how they are used with log-linear models via gazetteer features.

When using a CRF, or a log-linear model in general, on a task such as NER, one usually constructs features in the model which represent the dependencies between a word's NER label and local contextual properties of the word that are thought to influence the NER label, such as the word itself, its part-of-speech tag or some orthographic properties of the word. However, one sometimes encounters an entity which is difficult to identify using these local contextual cues alone because the entity has not been seen before. In these cases, a **gazetteer** or dictionary of possible entity identifiers is often useful. Such identifiers could be names of people, places, companies or other organisations. Using gazetteers one may define additional features in the model that represent the dependencies between a word's NER label and its presence in a particular gazetteer. Such **gazetteer features** are often highly informative, and their inclusion in the model should in principle result in higher model accuracy. We describe the gazetteers we use for the experiments in this chapter, and the features that encode dependencies upon them, in the next few sections.

To date gazetteers have been widely used in a variety of information extraction systems, including both rule-based systems and statistical models. In addition to lists of people names, locations, etc., recent work in the biomedical domain has utilised gazetteers of biological and genetic entities such as gene names (Finkel et al., 2005; McDonald and Pereira, 2005). In general gazetteers are thought to provide a useful source of external knowledge that is helpful when an entity cannot be identified from knowledge contained solely within the dataset used for training. However, some re-

search has questioned the usefulness of gazetteers (Krupka and Hausman, 1998; Morgan et al., 1995). Other work has supported the use of gazetteers in general but has found that lists of only moderate size are sufficient to provide most of the benefit (Mikheev et al., 1999). Therefore, to date the effective use of gazetteers for information extraction has in general been regarded as a "black art". As a result of this, in using gazetteer features as illustrative examples of the more general discriminative features we talked about earlier, we are also providing in this chapter a possible explanation for some of the conflicting results witnessed in the past when applying gazetteers to various information extraction tasks.

The material we present in this chapter is based on one of our conference papers (Smith and Osborne, 2006). While we were developing the ideas, Sutton et al. (2006), independently and in parallel, were following a similar theme. In their paper they identify general problems with gazetteer features and propose a solution similar to our LOP approach. They present results on NP-chunking in addition to NER, and provide a slightly more general approach. By contrast, we motivate the problem more thoroughly through analysis of the actual errors observed and through consideration of the success of other candidate solutions, such as traditional regularisation over feature subsets.

8.2 Experimental Setup

Before looking in more detail at the problems encountered when training models with highly discriminative features, we first describe the setup for the experiments in this chapter. We also provide results for the baseline models, with and without gazetteer features. Subsequent sections in this chapter discuss approaches which use, and extend, the models described here.

8.2.1 Task and Dataset

As we alluded to in the introduction to the chapter, we conduct our experiments on NER. This is mainly for illustrative purposes: gazetteer features are particularly good examples of the discriminative features whose effect we are trying to investigate in the chapter, and NER is a task particularly conducive to the use of gazetteers in helping to correctly label entities unseen in the training data. As before we use the CoNLL-2003 shared task English dataset (Kim Sang and Meulder, 2003) that we described in Chapter 3.

Gazetteer Name	Entry Type	# Entries
SUR-GAZ	People surnames	88,799
FEM-GAZ	female first names	4,275
MAL-GAZ	male first names	1,219
PLA-GAZ	place names	27,635
COMP-GAZ-1	company names	20,638
COMP-GAZ-2	company names	279,195
ORG-GAZ	organisation names[1]	425

Table 8.1: Properties of the seven gazetteers.

Bea	Anglican Church
Beata	Anglican Communion
Beatrice	Anti-Masonic Party
Beatris	Arab League
Beatriz	Arminian Baptist

Table 8.2: Extracts from the FEM-GAZ and ORG-GAZ gazetteers.

8.2.2 Gazetteers

For the experiments in this chapter we employ a total of seven gazetteers. These cover names of people, places and organisations. Table 8.1 gives details of the gazetteers, along with the names we will use to refer to the gazetteers in the chapter. Table 8.2 gives extracts from the FEM-GAZ and ORG-GAZ gazetteers. As can be seen from the table, the gazetteer entries may consist of single or multiple words.

8.2.3 Feature set

In Chapter 4 we described the STANDARD model for NER. For the experiments in this chapter we modify the STANDARD model by including an additional set of features defined using the gazetteers above. We call this modified model the STANDARD+G model, and call the extra features **gazetteer features**.

Our gazetteer features encode whether a particular word appears in a particular gazetteer. We divide the gazetteer features into two sets: **unlexicalised** and **lexi-**

[1] Names of organisations other than companies, such as government agencies, charities, etc.

Label predicates	Observation predicates
$s_t = s$	w_t is in gazetteer SUR-GAZ
	w_t is in gazetteer FEM-GAZ
	w_t is in gazetteer MAL-GAZ
	w_t is in gazetteer PLA-GAZ
	w_t is in gazetteer COM-GAZ-1
	w_t is in gazetteer COM-GAZ-2
	w_t is in gazetteer ORG-GAZ

Table 8.3: Feature templates for unlexicalised gazetteer features.

calised. The unlexicalised features model the dependency between a word's presence in a gazetteer and its NER label, irrespective of the word's identity. The lexicalised features, on the other hand, include the word's identity and so provide more refined word-specific modelling of the gazetteer-NER label dependency. The feature templates that define the unlexicalised and lexicalised gazetteer features are shown in Tables 8.3 and 8.4 respectively. Defining gazetteer features in this way is the typical method of representing gazetteer information in log-linear models. Note that some gazetteer entries involve multiple words where not all the words appear individually in the gazetteer. Consequently, the lexicalised gazetteer features are *not* simply determined by the word identity features.

In total there are 35 unlexicalised gazetteer features and $8,294$ lexicalised gazetteer features. Therefore, together with the $450,346$ features in the STANDARD model for NER, this gives a total of $458,675$ features in STANDARD+G model.

8.2.4 Baseline Results

Before exploring the overtraining effect that gazetteer feature can have in the next section, we first give baseline results for the STANDARD+G model. We include the results for the STANDARD model for comparison. Table 8.5 gives F scores for the two models, both unregularised and regularised models. Development set scores are included for completeness, and are referred to later in the chapter. The regularised models are trained with a zero-mean Gaussian prior, with the variance set using the development data.

From the table we see that the addition of the gazetteer features allows the STAN-

Label predicates	Observation predicates
$s_t = s$	$w_t = w$, w_t is in gazetteer SUR-GAZ
	$w_t = w$, w_t is in gazetteer FEM-GAZ
	$w_t = w$, w_t is in gazetteer MAL-GAZ
	$w_t = w$, w_t is in gazetteer PLA-GAZ
	$w_t = w$, w_t is in gazetteer COM-GAZ-1
	$w_t = w$, w_t is in gazetteer COM-GAZ-2
	$w_t = w$, w_t is in gazetteer ORG-GAZ

Table 8.4: Feature templates for lexicalised gazetteer features.

Model	Development		Test	
	Unreg.	Reg.	Unreg.	Reg.
STANDARD	88.21	89.86	81.60	83.97
STANDARD+G	89.19	90.40	83.10	84.70

Table 8.5: F scores for STANDARD and STANDARD+G models.

DARD+G model to outperform the STANDARD one, for both the unregularised and regularised models. In each case, the STANDARD+G model outperforms the STANDARD model at a significance level of $p < 0.02$. Therefore, the addition of the gazetteer features appears to lead to a positive improvement. However, as we alluded to in the introduction to this chapter, these results camouflage the fact that the gazetteer features introduce some negative effects. As such, the real benefit of including the gazetteer features in STANDARD+G is not fully realised.

8.3 Error Analysis of Gazetteer Features

We identify problems with the use of gazetteer features by considering test set labelling errors for both STANDARD and STANDARD+G. We use regularised models here as an illustration. Table 8.6 shows the number of sites (a site being a particular word at a particular position in a sentence) where labellings have improved, worsened or remained unchanged with respect to the gold-standard labelling with the addition of the gazetteer features. For example, the value in the top-left cell is the number of sites

STANDARD		STANDARD+G ✓	STANDARD+G ✗
	✓	44,945	160
	✗	228	1,333

Table 8.6: Test set errors.

where both the STANDARD and STANDARD+G models label words correctly.

The most interesting cell in the table is the top-right one, which represents sites where STANDARD is correctly labelling words but, with the addition of the gazetteer features, STANDARD+G mislabels them. At these sites, the addition of the gazetteer features actually worsens things. How well, then, could the STANDARD+G model do if it could somehow reduce the number of errors in the top-right cell? In fact, if it had correctly labelled those sites, a significantly higher test set F score of 90.36% would have been obtained. This potential upside suggests much could be gained from investigating ways of correcting the errors in the top-right cell. It is not clear whether there exists any approach that could correct *all* the errors in the top-right cell while simultaneously maintaining the state in the other cells, but approaches that are able to correct at least some of the errors should prove worthwhile.

On inspection of the sites where errors in the top-right cell occur, we observe that some of the errors occur in sequences where no words are in any gazetteer, so no gazetteer features are active for any possible labelling of these sequences. In other cases, the errors occur at sites where some of the gazetteer features appear to have dictated the label, but have made an incorrect decision. As a result of these observations, we classify the errors from the top-right cell of Table 8.6 into two types: **Type A** and **Type B**.

8.3.1 Type A Errors

We call Type A errors those errors that occur at sites where gazetteer features seem to have been *directly* responsible for the mislabelling. In these cases the gazetteer features effectively "over-rule" the other features in the model causing a mislabelling where the STANDARD model, without the gazetteer features, correctly labels the word.

An example of a Type A error is given in the sentence extract below:

```
about/O Healy/I-LOC
```

This is the labelling given by STANDARD+G. The correct label for `Healy` here is `I-PER`. The STANDARD model is able to decode this correctly as `Healy` appears in the training data with the `I-PER` label. The reason for the mislabelling by the STANDARD+G model is that `Healy` appears in both the gazetteer of place names and the gazetteer of person surnames. The feature encoding the gazetteer of place names with the `I-LOC` label has a parameter value (value of λ) of 4.20, while the feature encoding the gazetteer of surnames with the `I-PER` label has a parameter value of 1.96, and the feature encoding the word `Healy` with the `I-PER` label has a parameter value of 0.25. Although other features both at the word `Healy` and at other sites in the sentence contribute to the labelling of `Healy`, the influence of the first feature above dominates. So in this case the addition of the gazetteer features has confused things at test time.

8.3.2 Type B Errors

We call Type B errors those errors that occur at sites where the gazetteer features seem to have been only *indirectly* responsible for the mislabelling. In these cases the mislabelling appears to be more attributable to the non-gazetteer features, which are in some sense less expressive after being trained with the gazetteer features. Consequently, they are less able to decode words that they could previously label correctly.

An example of a Type B error is given in the sentence extract below:

```
Chanderpaul/O was/O
```

This is the labelling given by STANDARD+G. The correct labelling, given by STANDARD, is `I-PER` for `Chanderpaul`. In this case no words in the sentence (including the part not shown) are present in any of the gazetteers so no gazetteer features are active for any labelling of the sentence. Consequently, the gazetteer features do not contribute at all to the labelling decision. Non-gazetteer features in STANDARD+G are, however, unable to find the correct labelling for `Chanderpaul` when they previously could in the STANDARD model.

For both Type A and Type B errors it is clear that the gazetteer features in STANDARD+G are in some sense too “powerful” while the non-gazetteers features have become too “weak”. The question, then, is: can we train all the features in the model in a more sophisticated way so as to correct for these effects?

8.4 Feature Dependent Regularisation

One interpretation of the findings of our error analysis in the previous section is that the addition of the gazetteer features to the model is having an implicit over-regularising effect on the other features. Therefore, is it possible to adjust for this effect through more careful explicit regularisation using a prior? Can we directly regularise the gazetteer features more heavily and the non-gazetteer features less? We investigate this possibility in this section.

The STANDARD+G model is regularised by fitting a single Gaussian variance hyperparameter, σ^2, across all features. The optimal value for this single hyperparameter is 2025 ($\sigma = 45$). We now relax this single constraint by allocating a separate hyperparameter to different feature subsets, one for the gazetteer features (σ_{gaz}) and one for the non-gazetteer features ($\sigma_{non\text{-}gaz}$). The hope is that the differing subsets of features are best regularised using different prior hyperparameters. Clearly, by doing this we increase the search space significantly and encounter some of the problems we saw in Chapter 5. In order to make the search manageable, we constrain ourselves to three scenarios:

1. We fix $\sigma^2_{non\text{-}gaz}$, the variance for the non-gazetteer features, at the single variance optimum of 2025, and regularise the gazetteer features a little more.

2. We fix σ^2_{gaz}, the variance for the gazetteer features, at the single variance optimum of 2025, and regularise the non-gazetteer features a little less.

3. We simultaneously regularise the gazetteer features a little more than at the single variance optimum, and regularise the non-gazetteer features a little less.

Table 8.7 gives representative development set F scores for each of these three scenarios, with each scenario separated by a horizontal dividing line. We see that in general the results do not differ significantly from that of the single variance optimum. We conjecture that the reason for this is that the regularising effect of the gazetteer features on the non-gazetteer features is due to relatively subtle interactions during training that relate to the dependencies the features encode and how these dependencies overlap. Regularising different feature subsets by different amounts with a Gaussian prior does not directly address these interactions but instead just rather crudely penalises the magnitude of the parameter values of different feature sets to different degrees. Indeed this is true for any standardly formulated prior which makes independence assumptions

σ_{gaz}	$\sigma_{non-gaz}$	Development
42	45	90.40
40	45	90.30
45	46	90.39
45	50	90.38
44.8	45.2	90.41
43	47	90.35

Table 8.7: FDR development set F scores.

over the parameters[2]. It seems therefore that any solution to the regularising problem should come through more explicit restricting or removing of the interactions between gazetteer and non-gazetteer features during training.

8.5 LOPs

We may remove interactions between gazetteer and non-gazetteer features entirely by quarantining the gazetteer features and training them in a separate model. This allows the non-gazetteer features to be protected from the over-regularising effect of the gazetteer features. In order to decode taking advantage of the information contained in both models, we must combine the models in some way. To do this we use the logarithmic opinion pool that we introduced in Chapter 6.

In order to use a LOP for decoding we could employ a uniform distribution over the constituent models. However, we saw in Chapter 6 that in some cases manually tuning the weight distribution can give improved performance over a LOP with uniform weights. In this chapter we only construct LOPs consisting of two models in each case, one model with gazetteer features and one without. We can therefore efficiently tune the one free weight using the development set.

To construct models for the gazetteer and non-gazetteer features we first partition the feature set of the STANDARD+G model into the subsets outlined in Table 8.8. The **simple structural features** model *label-label* and *label-word* dependencies, while the **advanced structural features** include these features as well as those modelling *label-label-word* conjunctions. The **simple orthographic features** measure properties of a

[2]We talked about this assumption of independence between parameters when using a prior in section 2.3.1.

Feature Subset	Feature Type
S1	simple structural features
S2	advanced structural features
N	*n*-grams of words and POS tags
O	simple orthographic features
A	advanced orthographic features
G	gazetteer features

Table 8.8: STANDARD+G feature subsets.

LOP	Development	Test
STANDARD+g	90.40	84.70
S1G-STANDARD	**91.34**	**85.98**
S2G-STANDARD	91.32	85.59
S2NG-STANDARD	90.66	84.59
S2NOG-STANDARD	90.47	84.92
S2NOAG-STANDARD	90.56	84.78

Table 8.9: Regularised LOP F scores.

word such as capitalisation, presence of a digit, etc., while the **advanced orthographic properties** model the occurrence of prefixes and suffixes of varying length.

We create and train different models for the gazetteer features by adding different feature subsets to the gazetteer features. We regularise these models in the usual way using a Gaussian prior. In each case we then combine these models with the STANDARD model and decode under a LOP.

Table 8.9 gives results for LOP decoding for the different model pairs. Results for the STANDARD+G model are included in the first row for comparison. For each LOP the hyphen separates the two models comprising the LOP. So, for example, in the second row of the table we combine the gazetteer features with simple structural features in a model, train and decode with the STANDARD model using a LOP. The simple structural features are included so as to provide some basic support to the gazetteer features.

We see from Table 8.9 that the first two LOPs significantly outperform the regu-

LOP	LOP Weights
S1G-STANDARD	[0.39, 0.61]
S2G-STANDARD	[0.29, 0.71]
S2NG-STANDARD	[0.43, 0.57]
S2NOG-STANDARD	[0.33, 0.67]
S2NOAG-STANDARD	[0.39, 0.61]

Table 8.10: Regularised LOP weights.

larised STANDARD+G model (at a significance level of $p < 0.01$, on both the test and development sets). By training the gazetteer features separately we have avoided their over-regularising effect on the non-gazetteer features. This relies on training the gazetteer features with a relatively small set of other features. We can see this by reading down the table, below the top two rows. As more features are added to the model containing the gazetteer features, we obtain decreasing test set F scores because the advantage created from separate training of the features is increasingly lost.

Table 8.10 gives the corresponding weights for the LOPs in Table 8.9, which are set using the development data. We see that in every case the LOP allocates a smaller weight to the gazetteer features model than the non-gazetteer features model and in doing so restricts the influence that the gazetteer features have in the LOP's labelling decisions.

Table 8.11, similar to Table 8.6 earlier, shows test set labelling errors for the STANDARD model and the one of the LOPs. We take the S2G-STANDARD LOP here for illustration. We see from the table that the number of errors in the top-right cell shows a reduction of 29% over the corresponding value in Table 8.6. We have therefore reduced the number errors of the type we were targeting with our approach. The approach has also had the effect of reducing the number of errors in the bottom-right cell, which further improves model accuracy.

All the LOPs in Table 8.9 contain regularised constituent models. Table 8.12 gives test set F scores for the corresponding LOPs constructed from unregularised models. Although the scores are lower than those in Table 8.9, the S1G-STANDARD LOP still outperforms the *regularised* STANDARD+G model.

In summary, by training the gazetteer features and non-gazetteer features in separate models and decoding using a LOP, we are able to overcome the problems described

STANDARD		s2g-STANDARD LOP ✓	s2g-STANDARD LOP ✗
	✓	44,991	114
	✗	305	1,256

Table 8.11: Test set errors

LOP	Development	Test
S1G-STANDARD	**90.58**	**84.87**
S2G-STANDARD	90.70	84.28
S2NG-STANDARD	89.70	84.01
S2NOG-STANDARD	89.48	83.99
S2NOAG-STANDARD	89.40	83.70

Table 8.12: Unregularised LOP F scores.

in earlier sections and can achieve much higher accuracy. This shows that successfully deploying gazetteer features within log-linear models should involve careful consideration of restrictions on how features interact with each other, rather than simply considering the absolute values of feature parameters in isolation from each other.

8.6 More General Case: Gazetteer-Like Features

So far our discussion has focused on gazetteer features as illustrative examples of the general discriminative features that we talked about in the opening sections. To show that our findings are not specific to gazetteer features, we explore other sets of features with properties similar to those of gazetteer features. By applying similar treatment to these features during training we may be able harness their usefulness to a greater degree than is currently the case when training in a single model. So how might other discriminative features with similar properties be identified?

The task of identifying the optimal feature set partition for creation of models in the previous section is in general a hard problem because it relies on clustering the features based on their explanatory power relative to all other clusters. Both the number of

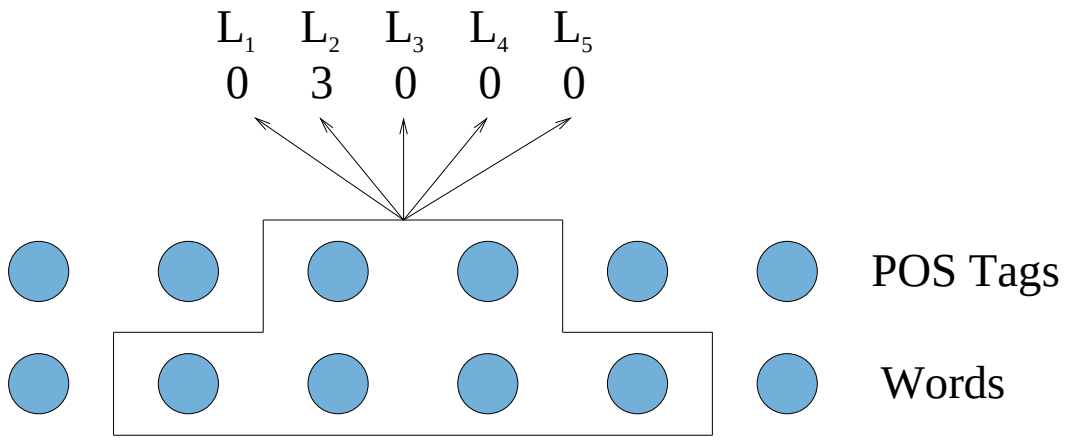

Figure 8.1: Pictorial depiction of a family n-ton feature for $n = 3$.

clusters and the distribution of features across the clusters must be determined.[3]

However, we may be able to solve the problem approximately. We have already found a partition manually, using our intuition about the problems of training highly discriminative features. Using this we have been able to arrive at a "reasonable" partition. One way forward, therefore, is to try to devise some heuristics that allow us to identify feature sets that have similar properties to those of discriminative set we have been using, i.e. the gazetteer features. In this section we consider three such heuristics. All of these heuristics are motivated by the observation that gazetteer features are both highly discriminative and generally very sparse. The heuristics lead to three candidate feature sets:

1. **Family Singleton Features (FSF)**. We define a feature **family** as a set of features that have the same conjunction of predicates defined on the observations. Hence they differ from each other only in the NER label that they encode. **Family singleton features** are features that have a count of 1 in the training data when all other members of that feature family have zero counts. These features have a flavour of gazetteer features in that they represent the fact that the conjunction of observation predicates they encode is highly predictive of the corresponding NER label, and that they are also very sparse.

2. **Family *n*-ton Features (FnF)**. These are features that have a count of n (greater than 1) in the training data when all other members of that feature family have

[3]This problem is very similar to the task of defining an optimal feature partition for the feature set experts that we saw in Chapter 6.

zero counts. They are similar to family singleton features, but exhibit gazetteer-like properties less and less as the value of n is increased because a larger value of n represents less sparsity. Figure 8.1 gives a pictorial representation of a single family of **family n-ton features**, for $n = 3$. In the diagram the feature family consists of features which all have the same observation part, represented by the box enclosing a local neighbourhood of words and POS tags, and a distribution of training data counts across possible labels that has all the mass on one label, in this case a count of 3 on label L_2.

3. **Loner Features (LF)**. These are features which occur with a low mean number of other active features in the training data. They are similar to gazetteer features in that, at the points where they occur, they are in some sense being relied upon more than most features to explain the data. Figure 8.2 gives a pictorial representation of where loner features will in general occur. The diagram shows the mean number of features active at different positions in a sentence in the training data, for the gold standard labelling. The loner features will typically occur at positions similar to the one marked, where only a small number of other features are active.

   To create loner feature sets we rank all features in the STANDARD+G model based on the mean number of other features they are observed within the training data, then we take subsets of increasing size. We present results for subsets of size 500, 1000, 5000 and 10000.

For each of these categories of features we add simple structural features (the S1 set from earlier), to provide basic structural support, and then train a regularised model. We also train a regularised model consisting of all features in STANDARD+G except the features from the category in question. We decode these model pairs under a LOP as described earlier.

Table 8.13 gives test set F scores for LOPs created from each of the categories of features above (with abbreviated names derived from the category names). The results show that for the **family singleton features** and each of the **loner feature** sets we obtain LOPs that significantly outperform the regularised STANDARD+G model ($p < 0.0002$ in every case). The **family n-ton features'** LOP does not do as well, but that is probably due to the fact that some of the features in this set have a large value of n and so behave much less like gazetteer features.

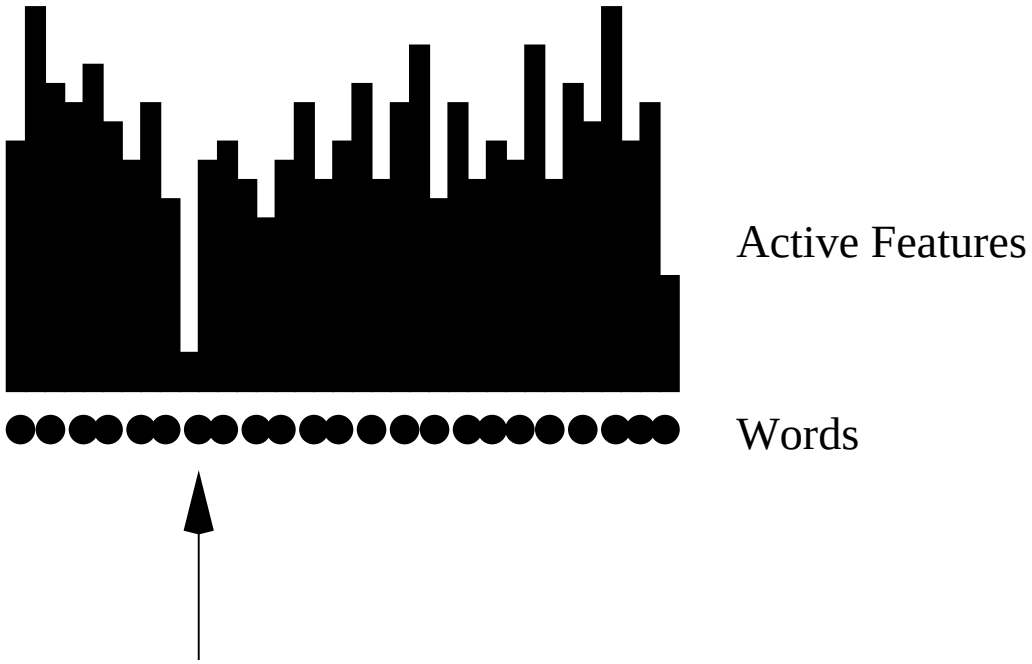

Figure 8.2: Pictorial depiction of number of features active at different positions within a sentence. The loner features are those features that occur only at places like the one marked, where only a small number of other features are active.

In summary, we obtain the same pattern of results using our quarantined training and LOP decoding method with these categories of features that we do with the gazetteer features. We conclude that the problems with gazetteer features that we have identified in this chapter are exhibited by general discriminative features with gazetteer feature-like properties, and our method is also successful with these more general features. Clearly, the heuristics that we have devised in this section are very simple, and it is likely that with more careful engineering better feature partitions can be found.

8.7 Summary

In this chapter we have identified and analysed negative effects that can be introduced to CRFs, and log-linear models in general, by the inclusion of highly discriminative features. We have used gazetteer features as a representative example of such a discriminative feature set. We have shown that these negative effects manifest themselves through errors that generally result from the model's over-dependence on the discriminative features for decision making. To overcome this problem a more careful treatment of these features is required during training. The solution we propose involves

LOP	Test Set
FSF	85.79
FnF	84.78
LF 500	85.80
LF 1000	85.70
LF 5000	85.77
LF 10000	85.62

Table 8.13: Regularised LOP F scores.

quarantining the highly discriminative features and training them separately from the other features in the model. Decoding is then undertaken using the logarithmic opinion pool framework that we introduced in Chapter 6. In fact, the LOP provides a natural way to handle the problem, with different constituent models for the different feature types. The method leads to much greater accuracy, and allows the power of discriminative features to be fully harnessed.

Towards the end of the chapter, we went on to identify other examples of highly discriminative feature sets (in addition to the gazetteer features we had been using though the chapter) by looking for features with gazetteer feature-like properties. We showed that similar problems, and our proposed solution, also exist for these more general discriminative feature sets.

Chapter 9

Conclusion

This thesis concerns regularisation techniques for conditional random fields (CRFs) within the domain of natural language processing (NLP). CRFs are conditional probabilistic models for structured labelling problems, and have been successfully applied to a number of different tasks in NLP and other domains. Despite their success, much work has supported the view that CRFs tend to overfit the data on which they are trained, significantly in some circumstances. As a result, effective application of CRFs requires the need for some form of regularisation. To date, conventional approaches to regularising CRFs have focused on the use of a Gaussian prior over the model parameters. However, in most cases fitting a Gaussian prior involves an element of trial-and-error, where one must search a potentially high-dimensional hyperparameter space.

In this thesis we address the overfitting problem in CRFs by investigating an alternative framework for CRF regularisation based on a form of CRF ensemble model. The main contributions of the thesis fall into three categories:

1. **Analysis of Conventional CRF Regularisation**. Our first contribution entails a thorough analysis of conventional regularisation techniques for CRFs, including some extensions which we propose. We start by considering regularisation with a prior distribution, and compare three families of priors. Our findings suggest that the Gaussian prior, although the most commonly used with CRFs in practice, is not clearly the most natural choice and other possibilities exist. This conclusion is contrary to the findings of previous work by Peng and McCallum (2004). We also consider how to apply regularisation with a prior to the features in a model in a feature-specific way. This differs from the usual application of a prior, where all parameters are regularised equally. We look at regularisation over subsets of features, investigating how such groupings may be defined,

and consider regularisation at the level of individual features, where each feature in the model is regularised to a different degree. Our general conclusion is that although improvements may be made with regularisation at a lower level of granularity, the benefits gained are typically offset by the complex search required to fit a larger number of hyperparameters. Lastly, we introduce a new, alternative approach to conventional regularisation by formulating a variant of the standard CRF model that has a regularising effect implicit in its form. We call this model the *inequality* CRF, and show that it can lead to significant performance improvement over a standard CRF regularised with a Gaussian prior, at the cost of only one additional hyperparameter.

2. **New Framework for CRF Regularisation**. Our second contribution forms the main focus of the thesis. Here, we introduce an alternative framework for CRF regularisation based on a form of ensemble model called a *logarithmic opinion pool* (LOP). The model combines a set of CRFs in a weighted product. We show that the LOP is a natural choice for a CRF ensemble due to the exponential form of the CRF distribution. We also demonstrate how the LOP satisfies an *ambiguity decomposition*, which motivates the need for the models in the LOP to be accurate and/or diverse. Such diversity may be created in a number of ways, including using the feature set, the training data and the training algorithm. We explore each of these possibilities.

   Our main result comes from using the feature set as a source of diversity. We show that by creating a set of CRF models based on an intuitively-motivated partition of the feature set, and combining them under a LOP with uniform weights, we may obtain a model which significantly outperforms an unregularised standard CRF that utilises the entire feature set, and is comparable in performance to a standard CRF regularised with a Gaussian prior. This means that the LOP approach with unregularised models represents a competitive alternative to conventional regularisation with a prior, but without the need to search a hyperparameter space.

   We also show that using the training set as a source of diversity is not as effective as use of the feature set. Employing CRFs trained from bagged training set samples, we obtain LOPs that do provide some improvement over an unregularised standard CRF, but are not competitive with a standard CRF regularised with a Gaussian prior.

Our LOP framework involves the multiplicative combination of CRFs in an ensemble model. A variant of this is the *linear opinion pool* (LIP), where models are combined additively. We investigate the properties of LIPs for CRFs and compare them to LOPs. We see that it is difficult to achieve efficient, accurate decoding of a LIP of CRFs, and that approximate decoding must be used. However, we also show how such approximation leads to decoding that is both more expensive than LOP decoding and results in lower performance. Our conclusion, therefore, is that a LOP is a more preferable combination method for CRFs than a LIP.

In addition to considering the feature set and training set as possible sources of diversity, we also investigate ways of encouraging diversity through the CRF training algorithm. In particular, we introduce the idea of *co-operative training* of CRFs, where parameters in different models interact during the training process. We encourage diversity between the models using a specially formulated objective function which includes a *diversity penalty* term. Our results demonstrate that the presence of the diversity term can lead to LOPs with improved performance. Our work in this area sets the scene for a number of future research possibilities.

3. **Applications of the Framework**. Our third contribution involves application of the LOP framework to overcome negative effects that a CRF may suffer under the standard training regime. Such effects can occur when highly discriminative features are included in the model. We identify *gazetteer* features as a representative example of such a discriminative feature set. We show that these negative effects manifest themselves through errors that generally result from the model's over-dependence on the discriminative features for decision making. To overcome this problem a more careful treatment of these features is required during training. We show that one solution to this problem involves quarantining the highly discriminative features and training them separately from the other features in the model. Decoding is then undertaken using the logarithmic opinion pool framework. In fact, the LOP provides a natural way to handle the problem, with different constituent models for the different feature types. We show that separate training and LOP decoding can lead to much greater accuracy, and allows the power of discriminative features to be fully harnessed. In particular, this method may provide for more effective use of gazetteers with CRF models

in the future. Lastly, we identify other examples of highly discriminative feature sets, in addition to gazetteer features, by identifying features with gazetteer feature-like properties. We show that similar problems also exist for these more general discriminative feature sets, and that the LOP approach may be used to address them.

In summary, this thesis demonstrates the potential for regularising CRFs in alternative ways to the conventional approaches in the literature. We present one such method in the thesis and explore its properties. We show that the approach can achieve performance levels for the CRF that are similar, or better, than those of traditional approaches but at lower cost.

9.1 Future Work

In each chapter of the thesis we have noted threads of work with the potential for future investigation. We now summarise some of the main possibilities:

1. In Chapter 6 (section 6.8), we looked at the possibility of creating LOPs with non-uniform weights. We described a procedure to find the optimal set of weights for a given set of constituent models. The approach was based on training the weights to maximise the log-likelihood of the training data. This log-likelihood was a function of the weights only, with the constituent models held fixed in their pre-trained states. We found that the procedure did not produce significant improvements over LOPs with a uniform weight distribution. We attributed this to the fact that for most of the feature set expert sets we have been using in the thesis, the optimal weight distribution is likely to be close to the uniform distribution because the experts are of roughly equal quality. It would be interesting to investigate this idea further and experiment with the behaviour of the algorithm in situations where the constituent models possess a greater differential in model quality. In such cases we would expect the LOP to downgrade the weight attached to the less accurate models, therefore partially removing these models from the LOP's labelling decisions.

   In addition to use of the log-likelihood in the objective function, it would also be possible to seek optimal weights based on an alternative measure. One possibility would be to try to maximise the most relevant criterion for the task. For example, for NER this could be F score.

2. In Chapter 7 we described a framework where parameters in different CRFs interact during training through a penalty term in the objective function that encourages diversity between the resulting models. Our results showed that the basic framework was successful and we were able to create LOPs that showed performance improvements over the LOPs of previous chapters, where the constituent models were trained independently. However, as we mentioned in Chapter 7, our findings really only represent a proof-of-concept for this body of work, and there are many avenues for future investigation. Possibilities include exploration of other forms of the co-operative training objective function, including more effective forms for the diversity penalty; the extension of the training algorithm to include LOP per-model weights as well as constituent model parameters; and examination of the behaviour of the framework with a wider range other expert sets.

3. The co-operative training framework of Chapter 7 takes as input a set of predefined constituent models, and trains the models interactively. For the experiments in the chapter we used the feature set expert sets that we had previously found to yield high performing LOPs. These experts sets were based on intuitively-motivated partitions of the feature set. In principle, however, we could generalise the co-operative training framework to include a feature induction facility as well as model training. Such a system would induce features in a similar way to single model feature induction for CRFs (McCallum, 2003), but across all models in an expert set simultaneously. The models would also be co-operatively trained within framework, with induction and training steps alternating. Clearly, design of such a system would involve a number of hard engineering decisions to maintain tractability, but the general idea represents an interesting avenue.

4. This thesis has concerned regularisation of CRFs in ways alternative to the conventional use of a prior distribution over the model parameters. However, it would be interesting to investigate the possibility of bridging the gap between the two approaches. For example, appropriate use of a Gaussian (or other) prior with a non-diagonal covariance matrix (where we do not assume model parameters are independent), may be able to simulate the behaviour of a LOP to some degree. Of course, application of such a prior would be problematic as it would involve specification of an even larger number of hyperparameters than with the

usual, diagonal case. Nonetheless, such a prior may shed light on the theoretical connections between priors and LOPs, and this would be very interesting.

Appendix A

Derivation of the CRF Probability Density Function

In Chapter 2 we defined a linear chain CRF as a model with the following distribution:

$$p(\mathbf{s} \mid \mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp\left(\sum_{t=1}^{T+1} \sum_{k=1}^{K} \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t)\right)$$

where $\mathbf{o}$ is a sequence of observations, $\mathbf{s}$ is the corresponding sequence of labels, the f_k are feature functions, the λ_k are model parameters associated with the f_k, t runs over the cliques in the sequence, and $Z(\mathbf{o})$ is a normalising function.

We now show that this general form of CRF distribution is the solution to a constrained optimisation problem, specifically a sequential entropy maximisation problem. The constraints in this problem have the form:

$$E_{p(\mathbf{s}|\mathbf{o})}[f_k] = C_k, \quad k = 1 \ldots K \tag{A.1}$$

where the C_k are a set of arbitrary constants. The optimisation problem can be stated as follows:

$$\begin{aligned} &\max_{p(\mathbf{s}|\mathbf{o})} \mathrm{H}[p(\mathbf{s} \mid \mathbf{o})] = -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o}) \\ &\text{s.t.} \quad E_{p(\mathbf{s}|\mathbf{o})}[f_k] = C_k, \quad k = 1 \ldots K \end{aligned} \tag{A.2}$$

We are therefore seeking the most general model (the one which maximises the sequential entropy) that obeys the constraints. In order to derive the general solution to

this problem, we appeal to Lagrangian methods. Good references in this area include the textbooks by Nocedal and Wright (1999) and Bertsekas (1999).

We first reformulate problem A.2 as a minimisation problem rather then a maximisation problem, and include an explicit normalisation constraint for the distribution $p(\mathbf{s}|\mathbf{o})$. This brings our setup into line with the standard form for the expression of constrained optimisation problems. Hence our problem trivially becomes:

$$\begin{aligned} &\min_{p(\mathbf{s}|\mathbf{o})} \; \mathrm{H}\left[p\left(\mathbf{s}\,|\,\mathbf{o}\right)\right] = \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o}) \\ &\text{s.t.} \quad E_{p(\mathbf{s}|\mathbf{o})}[f_k] = C_k, \;\; k = 1 \ldots K \\ &\qquad \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) - 1 = 0, \;\; \forall \mathbf{o} \text{ s.t. } \tilde{p}(\mathbf{o}) > 0 \end{aligned} \tag{A.3}$$

Note that we can multiply the last set of constraints by $\tilde{p}(\mathbf{o})$, for each $\mathbf{o}$, without violating any of the constraints in the set. We do this to make subsequent derivation easier. We also introduce a set of Lagrange multipliers α_k, each multiplier corresponding to a constraint in A.3, to form the Lagrangian:

$$\begin{aligned} \mathscr{L} = \quad & \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o}) \\ & + \sum_{k=1}^{K} \alpha_k \left(C_k - E_{p(\mathbf{s}|\mathbf{o})}[f_k]\right) \\ & + \sum_{\mathbf{o}} \mu_{\mathbf{o}} \tilde{p}(\mathbf{o}) \left(\sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) - 1\right) \end{aligned} \tag{A.4}$$

We now differentiate the Lagrangian with respect to the $p(\mathbf{s}|\mathbf{o})$ distribution. Let us consider a particular "element" of that distribution $p(\mathbf{s}'|\mathbf{o}')$. We must evaluate the derivative of the Lagrangian with respect to this element. Let us consider this term-by-term. The easiest term in the Lagrangian to differentiate is probably the last one. This differentiates to $\mu_{\mathbf{o}'} \tilde{p}(\mathbf{o}')$. Next let us evaluate the derivative of the first term in the Lagrangian. This is:

$$\begin{aligned} \frac{\partial}{\partial p(\mathbf{s}'|\mathbf{o}')} \left[\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o})\right] &= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \frac{\partial}{\partial p(\mathbf{s}'|\mathbf{o}')} \left[p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o})\right] \\ &= \tilde{p}(\mathbf{o}') \left[\log p(\mathbf{s}|\mathbf{o}') + 1\right] \end{aligned}$$

The derivative of the middle term in A.4 is a little more awkward to evaluate. Clearly the derivative of C_k vanishes and so we only need to look at the derivative of the expected value of f_k under the model. This derivative is:

$$\frac{\partial}{\partial p(\mathbf{s}'|\mathbf{o}')}\left[-\sum_{k=1}^{K}\alpha_k E_{p(\mathbf{s}|\mathbf{o})}[f_k]\right]$$

which expands to:

$$-\sum_{k=1}^{K}\alpha_k\frac{\partial}{\partial p(\mathbf{s}'|\mathbf{o}')}\left[\sum_{\mathbf{o}}\tilde{p}(\mathbf{o})\sum_{\mathbf{s}}p(\mathbf{s}|\mathbf{o})\sum_{t=1}^{T+1}f_k(\mathbf{s},\mathbf{o},t)\right]$$

with t running over the $T+1$ cliques in the sequence. We can see that this term differentiates to:

$$-\sum_{k=1}^{K}\alpha_k\tilde{p}(\mathbf{o}')\sum_{t=1}^{T+1}f_k\left(\mathbf{s}',\mathbf{o}',t\right)$$

Putting the derivatives of the three terms in A.4 together, we arrive at the derivative of the entire Lagrangian:

$$\frac{\partial\mathscr{L}}{\partial p(\mathbf{s}'|\mathbf{o}')}=\tilde{p}(\mathbf{o}')\left[1+\log p(\mathbf{s}'|\mathbf{o}')-\sum_{k=1}^{K}\alpha_k\sum_{t=1}^{T+1}f_k\left(\mathbf{s}',\mathbf{o}',t\right)+\mu_{\mathbf{o}'}\right]$$

We set this to zero, to give:

$$\tilde{p}(\mathbf{o}')\left[1+\log p(\mathbf{s}'|\mathbf{o}')-\sum_{k=1}^{K}\alpha_k\sum_{t=1}^{T+1}f_k\left(\mathbf{s}',\mathbf{o}',t\right)+\mu_{\mathbf{o}'}\right]=0 \tag{A.5}$$

We are only interested in the Lagrangian, and its derivatives, in that area of the space for which $\tilde{p}(\mathbf{o}')\neq 0$. Therefore, we can simplify Equation A.5 to:

$$1+\log p(\mathbf{s}'|\mathbf{o}')-\sum_{k=1}^{K}\alpha_k\sum_{t=1}^{T+1}f_k\left(\mathbf{s}',\mathbf{o}',t\right)+\mu_{\mathbf{o}'}=0$$

which gives:

$$\log p(\mathbf{s}'|\mathbf{o}')=-\mu_{\mathbf{o}'}-1+\sum_{k=1}^{K}\alpha_k\sum_{t=1}^{T+1}f_k\left(\mathbf{s}',\mathbf{o}',t\right)$$

Then exponentiating we get:

$$p(\mathbf{s}'|\mathbf{o}') = \exp(-\mu_{\mathbf{o}'} - 1) \exp\left(\sum_{k=1}^{K} \alpha_k \sum_{t=1}^{T+1} f_k(\mathbf{s}', \mathbf{o}', t)\right)$$

However, we know that $\sum_{\mathbf{s}'} p(\mathbf{s}'|\mathbf{o}') = 1$, so:

$$\exp(-\mu_{\mathbf{o}'} - 1) \sum_{\mathbf{s}'} \exp\left(\sum_{k=1}^{K} \alpha_k \sum_{t=1}^{T+1} f_k(\mathbf{s}', \mathbf{o}', t)\right) = 1$$

From this we can get the normalising function $Z(\mathbf{o}')$:

$$Z(\mathbf{o}') = \frac{1}{\exp(-\mu_{\mathbf{o}'} - 1)} = \sum_{\mathbf{s}'} \exp\left(\sum_{k=1}^{K} \alpha_k \sum_{t=1}^{T+1} f_k(\mathbf{s}', \mathbf{o}', t)\right)$$

Therefore, the distribution becomes:

$$p(\mathbf{s}'|\mathbf{o}') = \frac{1}{Z(\mathbf{o}')} \exp\left(\sum_{k=1}^{K} \alpha_k \sum_{t=1}^{T+1} f_k(\mathbf{s}', \mathbf{o}', t)\right)$$

Re-ordering the sums, removing the primes, re-labelling the Lagrange multipliers α_k (corresponding to the model parameters) to the more conventional λ_k, and noting that the cliques in a sequence are just label pairs (s_{t-1}, s_t), this probability density function can be re-stated as:

$$p(\mathbf{s}|\mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp\left(\sum_{t=1}^{T+1} \sum_{k=1}^{K} \lambda_k f_k(s_{t-1}, s_t, \mathbf{o})\right) \tag{A.6}$$

Hence we have derived the general form for a linear chain CRF distribution as the solution to a constrained sequential maximum entropy problem.

In the general probability density function given in A.6, the values of the λ_k depend on the original constants C_k in the constraints in A.1. Given a training data sample, the most natural choice for the constants C_k would be the empirical expected feature counts, calculated using the training data. Hence the constraints in A.1 would become:

$$E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] = 0, \;\; k = 1 \ldots K$$

With this specific choice for the constraints, the values of the model parameters λ_k correspond to the maximum likelihood CRF model.

Appendix B

The Inequality CRF

In this appendix we provide detailed derivations of some of the properties of the inequality CRF. Specifically, we derive the inequality CRF probability density function and the objective function used for training. The inequality CRF is introduced and discussed in Chapter 2.

B.1 Derivation of the Probability Density Function

In Chapter 5 we stated the optimisation problem for the inequality CRF as follows:

$$\begin{aligned} \max_{p(\mathbf{s}|\mathbf{o})} \; & \mathrm{H}[p(\mathbf{s}\,|\,\mathbf{o})] = -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o}) \\ \text{s.t.} \; & E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] - A_k \leq 0, \;\; k = 1 \ldots K \\ & E_{p(\mathbf{s}|\mathbf{o})}[f_k] - E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - B_k \leq 0, \;\; k = 1 \ldots K \end{aligned} \tag{B.1}$$

In order to derive the general solution to this problem, and therefore the general form for the probability density function of the inequality CRF, we appeal to Lagrangian methods. Our derivation follows very closely the derivation of the probability density function of a standard CRF, covered in Appendix A. However, in order to make each appendix self-contained, we will also show the details in full in this appendix. As mentioned in Appendix A, good references on Lagrangian methods include the textbooks by Nocedal and Wright (1999) and Bertsekas (1999).

We first reformulate the problem in B.1 as a minimisation problem rather then a maximisation problem, and include an explicit normalisation constraint for the distribution $p(\mathbf{s}|\mathbf{o})$. This brings our setup into line with the standard form for the expression of constrained optimisation problems. Hence our problem becomes:

$$\begin{aligned}
&\min_{p(\mathbf{s}|\mathbf{o})} \mathrm{H}[p(\mathbf{s}\,|\,\mathbf{o})] = \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o}) \qquad \text{(B.2)}\\
&\text{s.t.} \quad E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] - A_k \leq 0, \;\; k = 1 \ldots K\\
&\qquad E_{p(\mathbf{s}|\mathbf{o})}[f_k] - E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - B_k \leq 0, \;\; k = 1 \ldots K\\
&\qquad \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) - 1 = 0, \;\; \forall \mathbf{o} \text{ s.t. } \tilde{p}(\mathbf{o}) > 0
\end{aligned}$$

Note that we can multiply the last set of constraints by $\tilde{p}(\mathbf{o})$, for each $\mathbf{o}$, without violating any of the constraints in the set. Doing this makes subsequent derivation easier. Having done that, we introduce a set of Lagrange multipliers for each set of constraints in B.2, and form the Lagrangian:

$$\begin{aligned}
\mathscr{L} = \quad & \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o})\\
& + \sum_{k=1}^{K} \alpha_k \left(E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - E_{p(\mathbf{s}|\mathbf{o})}[f_k] - A_k \right)\\
& + \sum_{k=1}^{K} \beta_k \left(E_{p(\mathbf{s}|\mathbf{o})}[f_k] - E_{\tilde{p}(\mathbf{o},\mathbf{s})}[f_k] - B_k \right)\\
& + \sum_{\mathbf{o}} \mu_{\mathbf{o}} \tilde{p}(\mathbf{o}) \left(\sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) - 1 \right) \qquad \text{(B.3)}
\end{aligned}$$

We now differentiate the Lagrangian with respect to the $p(\mathbf{s}|\mathbf{o})$ distribution. Let us consider a particular "element" of that distribution $p(\mathbf{s}'|\mathbf{o}')$. We must evaluate the derivative of the Lagrangian with respect to this element. Let us consider this term-by-term. The easiest term in the Lagrangian to differentiate is probably the last one. This differentiates to $\mu_{\mathbf{o}'} \tilde{p}(\mathbf{o}')$. Next let us evaluate the derivative of the first term in the Lagrangian. This is:

$$\begin{aligned}
\frac{\partial}{\partial p(\mathbf{s}'|\mathbf{o}')} \left[\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o}) \right] &= \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \frac{\partial}{\partial p(\mathbf{s}'|\mathbf{o}')} \left[p(\mathbf{s}|\mathbf{o}) \log p(\mathbf{s}|\mathbf{o}) \right]\\
&= \tilde{p}(\mathbf{o}') \left[\log p(\mathbf{s}|\mathbf{o}') + 1 \right]
\end{aligned}$$

The derivatives of the middle two terms in B.3 are a little more awkward to evaluate, but are quite similar to each other so we only need evaluate one of them. Let us take the former of the two. Clearly the derivative of A_k vanishes and $p(\mathbf{s}'|\mathbf{o}')$ does not appear in the empirical expected value of f_k. Therefore we only need to look at the derivative of the expected value of f_k under the model. This derivative is:

$$\frac{\partial}{\partial p(\mathbf{s}'|\mathbf{o}')}\left[-\sum_{k=1}^{K}\alpha_k E_{p(\mathbf{s}|\mathbf{o})}[f_k]\right]$$

which expands to:

$$-\sum_{k=1}^{K}\alpha_k\frac{\partial}{\partial p(\mathbf{s}'|\mathbf{o}')}\left[\sum_{\mathbf{o}}\tilde{p}(\mathbf{o})\sum_{\mathbf{s}}p(\mathbf{s}|\mathbf{o})\sum_{t=1}^{T+1}f_k(\mathbf{s},\mathbf{o},t)\right]$$

with t running over the $T+1$ cliques in the sequence. We can see that this term differentiates to:

$$-\sum_{k=1}^{K}\alpha_k\tilde{p}(\mathbf{o}')\sum_{t=1}^{T+1}f_k\left(\mathbf{s}',\mathbf{o}',t\right)$$

It is obvious from this that the third term in the Lagrangian (with the β_k Lagrange multipliers) differentiates to:

$$\sum_{k=1}^{K}\beta_k\tilde{p}(\mathbf{o}')\sum_{t=1}^{T+1}f_k\left(\mathbf{s}',\mathbf{o}',t\right)$$

Putting all these individual derivatives together, we arrive at the derivative of the entire Lagrangian:

$$\frac{\partial\mathscr{L}}{\partial p(\mathbf{s}'|\mathbf{o}')}=\tilde{p}(\mathbf{o}')\left[1+\log p(\mathbf{s}'|\mathbf{o}')-\sum_{k=1}^{K}(\alpha_k-\beta_k)\sum_{t=1}^{T+1}f_k\left(\mathbf{s}',\mathbf{o}',t\right)+\mu_{\mathbf{o}'}\right]$$

We set this to zero, to give:

$$\tilde{p}(\mathbf{o}')\left[1+\log p(\mathbf{s}'|\mathbf{o}')-\sum_{k=1}^{K}(\alpha_k-\beta_k)\sum_{t=1}^{T+1}f_k\left(\mathbf{s}',\mathbf{o}',t\right)+\mu_{\mathbf{o}'}\right]=0 \tag{B.4}$$

We are only interested in the Lagrangian, and its derivatives, in that area of the space for which $\tilde{p}(\mathbf{o}')\neq 0$. Therefore, we can simplify B.4 to:

$$1+\log p(\mathbf{s}'|\mathbf{o}')-\sum_{k=1}^{K}(\alpha_k-\beta_k)\sum_{t=1}^{T+1}f_k\left(\mathbf{s}',\mathbf{o}',t\right)+\mu_{\mathbf{o}'}=0$$

which gives:

$$\log p(\mathbf{s}'|\mathbf{o}') = -\mu_{\mathbf{o}'} - 1 + \sum_{k=1}^{K} (\alpha_k - \beta_k) \sum_{t=1}^{T+1} f_k(\mathbf{s}', \mathbf{o}', t)$$

Then exponentiating we get:

$$p(\mathbf{s}'|\mathbf{o}') = \exp(-\mu_{\mathbf{o}'} - 1) \exp\left(\sum_{k=1}^{K} (\alpha_k - \beta_k) \sum_{t=1}^{T+1} f_k(\mathbf{s}', \mathbf{o}', t) \right)$$

However, we know that $\sum_{\mathbf{s}'} p(\mathbf{s}'|\mathbf{o}') = 1$, so:

$$\exp(-\mu_{\mathbf{o}'} - 1) \sum_{\mathbf{s}'} \exp\left(\sum_{k=1}^{K} (\alpha_k - \beta_k) \sum_{t=1}^{T+1} f_k(\mathbf{s}', \mathbf{o}', t) \right) = 1$$

From this we can get the normalising function $Z(\mathbf{o}')$:

$$Z(\mathbf{o}') = \frac{1}{\exp(-\mu_{\mathbf{o}'} - 1)} = \sum_{\mathbf{s}'} \exp\left(\sum_{k=1}^{K} (\alpha_k - \beta_k) \sum_{t=1}^{T+1} f_k(\mathbf{s}', \mathbf{o}', t) \right)$$

Therefore, we arrive at the general form for the probability density function for the inequality CRF:

$$p(\mathbf{s}'|\mathbf{o}') = \frac{1}{Z(\mathbf{o}')} \exp\left(\sum_{k=1}^{K} (\alpha_k - \beta_k) \sum_{t=1}^{T+1} f_k(\mathbf{s}', \mathbf{o}', t) \right)$$

Re-ordering the sums, removing the primes and noting that the cliques in a sequence are just label pairs (s_{t-1}, s_t), this probability density function can be re-stated as:

$$p(\mathbf{s}|\mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp\left(\sum_{t=1}^{T+1} \sum_{k=1}^{K} (\alpha_k - \beta_k) f_k(s_{t-1}, s_t, \mathbf{o}) \right)$$

The theory behind the use of Lagrangian methods (Nocedal and Wright, 1999) dictates that when the constraints are expressed as in B.2, the Lagrange multipliers are all non-negative. Therefore we have conditions on the α_k and β_k:

$$\alpha_k \geq 0, \;\; \beta_k \geq 0, \;\; k = 1 \ldots K$$

In addition, the α_k and β_k also satisfy the Karush-Kuhn-Tucker (KKT) conditions (Nocedal and Wright, 1999). These conditions are:

$$\begin{aligned}\alpha_k \left(E_{\tilde{p}}[f_k] - E_p[f_k] - A_k\right) &= 0, \quad k = 1 \ldots K \\ \beta_k \left(E_p[f_k] - E_{\tilde{p}}[f_k] - B_k\right) &= 0, \quad k = 1 \ldots K\end{aligned}$$

Because the inequality CRF model parameters (the α_k and β_k) are non-negative, the training process involves a bounded optimisation problem. To handle this we use the bounded limited memory variable metric (BLMVM) algorithm (Benson and More, 2001), which is implemented via special routines in the TAO libraries (see Chapter 3).

B.2 Derivation of the Objective Function

Following Lagrangian methods, we derive an objective function for the inequality CRF by substituting the values found in the previous section for the Lagrange multipliers, back into the Lagrangian. To simplify the orthography, let us denote the exponent in the probability density function for the inequality CRF by E, so:

$$E\left(\mathbf{o}, \mathbf{s}\right) = \sum_{t=1}^{T+1} \sum_{k=1}^{K} \left(\alpha_k - \beta_k\right) f_k\left(\mathbf{s}, \mathbf{o}, t\right)$$

Using this, the Lagrangian becomes:

$$\begin{aligned}\mathscr{L} = \quad & \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \frac{e^{E(\mathbf{o},\mathbf{s})}}{Z\left(\mathbf{o}\right)} \left[E\left(\mathbf{o}, \mathbf{s}\right) - \log Z\left(\mathbf{o}\right)\right] \\ & + \sum_{k=1}^{K} \alpha_k \left[\sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o}, \mathbf{s}) \sum_{t=1}^{T+1} \sum_{k=1}^{K} f_k\left(\mathbf{s}, \mathbf{o}, t\right) - \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \frac{e^{E(\mathbf{o},\mathbf{s})}}{Z\left(\mathbf{o}\right)} \sum_{t=1}^{T+1} \sum_{k=1}^{K} f_k\left(\mathbf{s}, \mathbf{o}, t\right) - A_k\right] \\ & + \sum_{k=1}^{K} \beta_k \left[\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \frac{e^{E(\mathbf{o},,\ k=1 \ldots K\mathbf{s})}}{Z\left(\mathbf{o}\right)} \sum_{t=1}^{T+1} \sum_{k=1}^{K} f_k\left(\mathbf{s}, \mathbf{o}, t\right) - \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o}, \mathbf{s}) \sum_{t=1}^{T+1} \sum_{k=1}^{K} f_k\left(\mathbf{s}, \mathbf{o}, t\right) - B_k\right] \\ & + 0\end{aligned}$$

Collecting terms and simplifying gives:

$$\begin{aligned}\mathscr{L} = \quad & \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \frac{e^{E(\mathbf{o},\mathbf{s})}}{Z(\mathbf{o})} \left[E(\mathbf{o},\mathbf{s}) - \log Z(\mathbf{o})\right] \\ & - \sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \sum_{\mathbf{s}} \frac{e^{E(\mathbf{o},\mathbf{s})}}{Z(\mathbf{o})} E(\mathbf{o},\mathbf{s}) \\ & + \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) E(\mathbf{o},\mathbf{s}) \\ & - \sum_{k=1}^{K} \alpha_k A_k - \sum_{k=1}^{K} \beta_k B_k\end{aligned}$$

Further simplifying leads to:

$$\mathscr{L} = -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \log Z(\mathbf{o}) + \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) E(\mathbf{o},\mathbf{s}) - \sum_{k=1}^{K} \alpha_k A_k - \sum_{k=1}^{K} \beta_k B_k \tag{B.5}$$

Actually, the first two terms on the right-hand side are just the log-likelihood under the inequality CRF. To see this, consider the definition of the log-likelihood:

$$\begin{aligned}LL(\alpha,\beta) &= \log \left[\prod_{\mathbf{o},\mathbf{s}} p(\mathbf{s}|\mathbf{o})^{\tilde{p}(\mathbf{o},\mathbf{s})}\right] \\ &= \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \log \left[\frac{e^{E(\mathbf{o},\mathbf{s})}}{Z(\mathbf{o})}\right] \\ &= \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) \left[E(\mathbf{o},\mathbf{s}) - \log Z(\mathbf{o})\right] \\ &= -\sum_{\mathbf{o}} \tilde{p}(\mathbf{o}) \log Z(\mathbf{o}) + \sum_{\mathbf{o},\mathbf{s}} \tilde{p}(\mathbf{o},\mathbf{s}) E(\mathbf{o},\mathbf{s})\end{aligned}$$

Comparing this to B.5 we see that the Lagrangian, and therefore the objective function, is just:

$$\mathscr{L} = LL(\alpha,\beta) - \sum_{k=1}^{K} \alpha_k A_k - \sum_{k=1}^{K} \beta_k B_k$$

This is the expression for the objective function given in Chapter 5, Equation 5.22.

Bibliography

Altun, Y., Johnson, M., and Hofmann, T. (2003). Investigating Loss Functions and Optimization Methods for Discriminative Learning of Label Sequences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2003), July 11-12, Sapporo, Japan.*

Asahara, M. and Matsumoto, Y. (2003). Japanese Named Entity Extraction with Redundant Morphological Analysis. In *Proceedings of the Human Technology Conference - North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003), May 27 - June 1, Edmonton, Canada*, pages 8–15.

Balay, S., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H. (2004). PETSc Users Manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, Illinois, United States.

Balay, S., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H. (2001). PETSc Web page. http://www.mcs.anl.gov/petsc.

Benediktsson, J. A. and Swain, P. H. (1992). Consensus Theoretic Classification Methods. *IEEE Transactions on Systems, Man and Cybernetics*, 22(4):688–704.

Benello, J., Mackie, A. W., and Anderson, J. A. (1989). Syntactic Category Disambiguation with Neural Networks. *Computer Speech and Language*, 3:203–217.

Benson, S., McInnes, L. C., More, J. J., and Sarich, J. (2002). *TAO Users Manual*. Argonne National Laboratory, Illinois, United States.

Benson, S. and More, J. (2001). A Limited Memory Variable Metric Method for Bound Constrained Minimization. Technical Report ANL/MCS-P909-0901, Argonne National Laboratory, Illinois, United States.

Benson, S. J., McInnes, L. C., Moré, J., and Sarich, J. (2005). TAO User Manual (Revision 1.8). Technical Report ANL/MCS-TM-242, Argonne National Laboratory, Illinois, United States.

Bertsekas, D. P. (1999). *Nonlinear Programming*. Athena Scientific, 2nd edition.

Besag, J. (1974). Spatial Interaction and the Statistical Analysis of Lattice Systems. *Journal of the Royal Statistical Society*, Series B(36):192–236.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Bordley, R. F. (1982). A Multiplicative Formula for Aggregating Probability Assessments. *Management Science*, 28:1137–1148.

Borthwick, A. (1999). *A Maximum Entropy Approach to Named Entity Recognition*. PhD thesis, New York University, United States.

Bottou, L. (1991). *Une Approche Théorique de l'Apprentissage Connexionniste: Applications á la Reconnaissance de la Parole*. PhD thesis, Université de Paris XI, 91405 Orsay Cedex, France.

Bourlard, H. and Wellekens, C. J. (1990). Links between Markov Models and Multilayer Perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(12):1167–1178.

Brill, E. (1994). Some Advances in Transformation-Based Part of Speech Tagging. In *Proceedings of the 12th National Conference on Artificial Intelligence, July 31 - August 4, Seattle, Washington, United States*, pages 722–727.

Brill, E. (1995). Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, 21:543–565.

Brown, A. D. and Hinton, G. E. (2000). Products of Hidden Markov Models. Technical report, University College, London, United Kingdom.

Brown, G. and Wyatt, J. (2003). Negative Correlation Learning and the Ambiguity Family of Ensemble Methods. In *Proceedings of the International Workshop on Multiple Classifier Systems (LNCS 2709), June 11-13, Guildford, Surrey, United Kingdom*, pages 266–275.

Brown, G., Yao, X., Wyatt, J., Wersing, H., and Sendhoff, B. (2002). Exploiting Ensemble Diversity for Automatic Feature Extraction. In *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02), November 18-22, Singapore*.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. Y. (1995). A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific Computing*, 16(6):1190–1208.

Carlin, B. P. and Louis, T. A. (2000). *Bayes and Empirical Bayes Methods for Data Analysis*. Chapman and Hall, 2nd edition.

Carney, J. and Cunningham, P. (1999). Tuning Diversity in Bagged Neural Network Ensembles. Technical Report TCD-CS-1999-44, Trinity College, Dublin, Ireland.

Chen, S. and Rosenfeld, R. (2000). A Survey of Smoothing Techniques for ME Models. In *IEEE Transactions on Speech and Audio Processing*, volume 8(1), pages 37–50.

Chen, S. F. and Rosenfeld, R. (1999). A Gaussian Prior for Smoothing Maximum Entropy Models. Technical Report CMU-CS-99-108, Computer Science Department, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States.

Chieu, H. L. and Ng, H. T. (2003). Named Entity Recognition with a Maximum Entropy Approach. In *Proceedings of the 7th Conference on Computational Natural Language Learning (CoNLL-2003), May 31 - June 1, Edmonton, Canada*.

Clark, S. and Curran, J. R. (2004). Parsing the WSJ using CCG and Log-Linear Models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004), July 21-26,Barcelona, Spain*.

Cohn, T. (2006). *Scaling Conditional Random Fields for Natural Language Processing*. PhD thesis, University of Melbourne, Australia.

Cohn, T. and Blunsom, P. (2005). Semantic Role Labelling with Tree Conditional Random Fields. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL-2005) Shared Task, June 29-30, Ann Arbor, Michigan, United States*.

Collins, M. (2002). Ranking Algorithms for Named-Entity Extraction: Boosting and the Voted Perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002), July 6-12, Philadelphia, Pennsylvania, United States*, pages 489–496.

Culotta, A., Kulp, D., and McCallum, A. (2005). Gene Prediction with Conditional Random Fields. Technical Report UM-CS-2005-028, University of Massachusetts, Amherst, Massachusetts, United States.

Curran, J. and Clark, S. (2003). Language Independent NER using a Maximum Entropy Tagger. In *Proceedings of the 7th Conference on Computational Natural Language Learning (CoNLL-2003), May 31 - June 1, Edmonton, Canada*.

Daelemanns, W., Zavrel, J., Berck, P., and Gillis, S. (1996). MBT: A Memory-Based Part-of-Speech Tagger-Generator. In *Proceedings of the 4th Workshop on Very Large Corpora, August 4, Copenhagen, Denmark*, pages 14–27.

Daelemans, W., Zavrel, J., van den Bosch, A., and van der Sloot, K. (2002). MBT: Memory Based Tagger, Version 1.0, Reference Guide. ILK Technical Report 02-09, Tilburg, The Netherlands.

Darroch, J. N. and Ratcliff, D. (1972). Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, 43(5):1470–1480.

Della Pietra, S., Della Pietra, V., and Lafferty, J. (1997). Inducing Features of Random Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393.

Farmakiotou, D., Karkaletsis, V., Koutsias, J., Sigletos, G., Spyropoulos, C. D., and Stamatopoulos, P. (2000). Rule-Based Named Entity Recognition for Greek Financial Texts. In *Proceedings of the Workshop on Computational Lexicography and Multimedia Dictionaries (COMLEX 2000), September 22-27, Patras, Greece*, pages 75–78.

Finkel, J., Dingare, S., Manning, C. D., Nissim, M., Alex, B., and Grover, C. (2005). Exploring the Boundaries: Gene and Protein Identification in Biomedical Text. *BMC Bioinformatics*, 6(S5).

Fletcher, R. and Reeves, C. M. (1964). Function Minimization by Conjugate Gradients. *The Computer Journal*, 7:149–154.

Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. (2003). Named Entity Recognition through Classifier Combination. In *Proceedings of the 7th Conference on Computational Natural Language Learning (CoNLL-2003) Shared Task, May 31 - June 1, Edmonton, Canada*.

Freitag, D. and McCallum, A. (2000). Information Extraction with HMM Structures Learned by Stochastic Optimization. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000), July 30 - August 3, Austin, Texas, United States*.

Gelman, A., Clarin, J. B., Stern, H. S., and Rubin, D. B. (2003). *Bayesian Data Analysis*. Chapman and Hall, 2nd edition.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4:1–58.

Gillick, L. and Cox, S. (1989). Some Statistical Issues in the Comparison of Speech Recognition Algorithms. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP-89), May 23-26, Glasgow, United Kingdom*, volume 1, pages 532–535.

Goodman, J. (2004). Exponential Priors for Maximum Entropy Models. In *Proceedings of the Human Technology Conference - North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004), May 2-7, Boston, Massachusetts, United States*.

Gregory, M. L. and Altun, Y. (2004). Using Conditional Random Fields to Predict Pitch Accents in Conversational Speech. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004), July 21-26, Barcelona, Spain*.

Grishman, R. and Sundheim, B. (1996). Message Understanding Conference - 6: A Brief History. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96), August 5-9, Copenhagen, Denmark*, pages 466–471.

Hammersley, J. M. and Clifford, P. (1971). Markov Fields on Finite Graphs and Lattices. Unpublished manuscript.

Hansen, V. and Krogh, A. (2000). A General Method for Combining Predictors on Protein Secondary Structure Prediction. In *Proceedings of the 1st International Conference on Artificial Neural Networks in Medicine and Biology (ANNIMAB 2000), May 13-16, Göteborg, Sweden*, pages 259–264.

He, X., Zemel, R., and Carreira-Perpin, M. . (2004). Multiscale Conditional Random Fields for Image Labelling. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), June 27 - July 2, Washington, D.C., United States*.

Heskes, T. (1998). Selecting Weighting Factors in Logarithmic Opinion Pools. In *Proceedings of the Conference on Advances in Neural Information Processing Systems 10, December 1-3, Denver, Colorado, United States*.

Hinton, G. E. (1999). Products of Experts. In *Proceedings of the 9th International Conference on Neural Networks (ICANN 99), September 7-10, University of Edinburgh, Edinburgh, United Kingdom*, volume 1, pages 1–6.

Hinton, G. E. (2002). Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14 (8):1771–1800.

Jelinek, F. (1985). *Markov Source Modeling of Text Generation*, volume E21, pages 569–598. Kluwer Academic Publishers.

Kazama, J. and Tsujii, J. (2003). Evaluation and Extension of Maximum Entropy Models with Inequality Constraints. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2003), July 11-12, Sapporo, Japan*.

Kim, J.-H. and Woodland, P. C. (2000). A Rule-Based Named Entity Recognition System for Speech Input. In *Proceedings of the 6th International Conference of Spoken Language Processing (Interspeech 2000), October 16-20, Beijing, China*, pages 521–524.

Kim Sang, E. F. T. (2002). Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the 6th Conference on Computational Natural Language Learning (CoNLL-2002), August 31 - September 1, Taipei, Taiwan*.

Kim Sang, E. F. T. and Buchholz, S. (2000). Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of the 4th Conference on Computational Language Learning (CoNLL-2000), September 13-14, Lisbon, Portugal.*

Kim Sang, E. F. T. and Meulder, F. D. (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the 7th Conference on Computational Natural Language Learning (CoNLL-2003), May 31 - June 1, Edmonton, Canada*.

Klein, D. and Manning, C. D. (2002). Conditional Structure Versus Conditional Estimation in NLP Models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2002), July 6-7, University of Pennsylvania, Philadelphia, Pennsylvania, United States.*

Klein, D., Smarr, J., Nguyen, H., and Manning, C. D. (2003). Named Entity Recognition with Character-Level Models. In *Proceedings of the 7th Conference on Computational Natural Language Learning (CoNLL-2003), May 31 - June 1, Edmonton, Canada*.

Klein, S. and Simmons, R. F. (1963). A Computational Approach to Grammatical Coding of English Words. *Journal of the Association for Computing Machinery*, 10:334–337.

Krogh, A. and Riis, S. K. (1999). Hidden Neural Networks. *Neural Computation*, 11(2):541–563.

Krogh, A. and Vedelsby, J. (1995). Neural Network Ensembles, Cross Validation, and Active Learning. In *Proceedings of the Conference on Advances in Neural Information Processing Systems 7, November 27-30, Denver, Colorado, United States*.

Kruengkrai, C., Sornlertlamvanich, V., and Isahara, H. (2006). A Conditional Random Field Framework for Thai Morphological Analysis. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006), May 24-26, Genoa, Italy*.

Krupka, G. R. and Hausman, K. (1998). Isoquest Inc: Description of the NetOwl (TM) Extractor System as used for MUC-7. In *Proceedings of the 7th Message Understanding Conference (MUC-7), April 29 - May 1, Fairfax, Virginia, United States*.

Kudo, T., Yamamoto, K., and Matsumoto, Y. (2004). Applying Conditional Random Fields to Japanese Morphological Analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004), July 21-26, Barcelona, Spain*.

Kupiec, J. (1992). Robust Part-of-Speech Tagging Using a Hidden Markov Model. *Computer Speech and Language*, 6:225–242.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001), June 28 - July 1, Williams College, Massachusetts, United States*.

Lee, Y.-H., Kim, M.-Y., and Lee, J.-H. (2005). Chunking Using Conditional Random Fields in Korean Texts. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP 2005), October 11-13, Jeju Island, South Korea*, pages 155–164.

Li, W. and McCallum, A. (2003). Rapid Development of Hindi Named Entity Recognition Using Conditional Random Fields and Feature Induction. *ACM Transactions on Asian Language Information Processing (TALIP)*, 2(3):290–294.

Liu, Y. and Yao, X. (1999). Ensemble Learning via Negative Correlation. *Neural Networks*, 12(10):1399–1404.

Malouf, R. (2002). A Comparison of Algorithms for Maximum Entropy Parameter Estimation. In *Proceedings of the 6th Conference on Computational Natural Language Learning (CoNLL-2002), August 31 - September 1, Taipei, Taiwan*.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Mayraz, G. and Hinton, G. E. (2002). Recognizing Handwritten Digits Using Hierarchical Products of Experts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24 (2):189–197.

McCallum, A. (2003). Efficiently Inducing Features of Conditional Random Fields. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI-2003), August 8-10, Acapulco, Mexico*.

McCallum, A., Freitag, D., and Pereira, F. (2000). Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000), June 29 - July 2, Stanford University, California, United States*.

McCallum, A. and Li, W. (2003). Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. In *Proceedings of the 7th Conference on Computational Natural Language Learning (CoNLL-2003), May 31 - June 1, Edmonton, Canada.*

McCallum, A., Rohanimanesh, K., and Sutton, C. (2003). Dynamic Conditional Random Fields for Jointly Labeling Multiple Sequences. In *Proceedings of the NIPS-2003 Workshop on Syntax, Semantics and Statistics, December 11-13, Vancouver, Canada*.

McCallum, A. and Wellner, B. (2004). Conditional Models of Identity Uncertainty with Application to Noun Coreference. In *Advances in Neural Information Processing Systems 16, December 13-16, Vancouver, Canada*.

McDonald, R. and Pereira, F. (2005). Identifying Gene and Protein Mentions in Text Using Conditional Random Fields. *BMC Bioinformatics*, 6(Supplement 1).

Melville, P. and Mooney, R. J. (2005). Creating Diversity in Ensembles Using Artificial Data. *Information Fusion*, 6(1):99–111.

Merialdo, B. (1994). Tagging English Text with a Probabilistic Model. *Computational Linguistics*, 20:155–171.

Mikheev, A., Moens, M., and Grover, C. (1999). Named Entity Recognition without Gazetteers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL'99), June 8-12, Bergen, Norway*, pages 1–8.

Minka, T. P. (2001). Expectation Propagation for Approximate Bayesian Inference. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI 2001), August 2-5, Seattle, Washington, United States*.

Minka, T. P. (2004). Power EP. Technical Report MSR-TR-2004-149, Microsoft Research, Cambridge, United Kingdom.

Molina, A. and Pla, F. (2002). Shallow Parsing using Specialized HMMs. *Journal of Machine Learning Research*, 2:595–613.

Morgan, R., Garigliano, R., Callaghan, P., Poria, S., Smith, M., Urbanowicz, A., Collingham, R., Costantino, M., and Cooper, C. (1995). Description of the LOLITA System as used in MUC-6. In *Proceedings of the 6th Message Understanding Conference (MUC-6), November 6-8, Columbia, Maryland, United States*.

Nadas, A. (1983). A Decision Theoretic Formulation of a Training Problem in Speech Recognition and a Comparison of Training by Unconditional Versus Conditional Maximum Likelihood. *IEEE Transactions on Acoustics, Speech and Signal Processing [see also IEEE Transactions on Signal Processing]*, 31(4):814–817.

Ng, A. Y. and Jordan, M. I. (2002). On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14, December 9-14, Vancouver, Canada*.

Nocedal, J. (1980). Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35:773–782.

Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*. Springer-Verlag, 1st edition.

Opitz, D. (1999). Feature Selection for Ensembles. In *16th National Conference on Artificial Intelligence (AAAI), July 18-22, Orlando, Florida*, pages 379–384.

Oza, N. C. and Tumer, K. (2001). Input Decimation Ensembles: Decorrelation through Dimensionality Reduction. In *Proceedings of the 2nd International Workshop on Multiple Classifier Systems*, pages 238–247. Springer-Verlag.

Peng, F., Feng, F., and McCallum, A. (2004). Chinese Segmentation and New Word Detection using Conditional Random Fields. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004), August 23-27, Geneva, Switzerland*.

Peng, F. and McCallum, A. (2004). Accurate Information Extraction from Research Papers using Conditional Random Fields. In *Proceedings of the Human Technology Conference - North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004), May 2-7, Boston, Massachusetts, United States*.

Perrone, M. P. and Cooper, L. N. (1993). When Networks Disagree: Ensemble Methods for Hybrid Neural Networks. In Mammone, R. J., editor, *Neural Networks for Speech and Image Processing*, pages 126–142. Chapman-Hall.

Pinto, D., McCallum, A., Wei, X., and Croft, W. B. (2003). Table Extraction Using Conditional Random Fields. In *Proceedings of the 26th Annual International ACM SIGIR Conference, July 28 - August 1, Toronto, Canada*.

Polak, E. and Ribière, G. R. (1969). Note sur la Convergence de Directions Conjugées. *Francaise Informat Recherche Operationelle*, 3(16):35–43.

Qi, Y., Szummer, M., and Minka, T. P. (2005). Bayesian Conditional Random Fields. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTATS), January 6-8, The Savannah Hotel, Barbados*.

Quattoni, A., Collins, M., and Darrel, T. (2005). Conditional Random Fields for Object Recognition. In *Advances in Neural Information Processing Systems 17, December 13-16, Vancouver, Canada*.

Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–285.

Ramshaw, L. A. and Marcus, M. P. (1995). Text Chunking Using Transformation-Based Learning. In *Proceedings of the 3rd Annual Workshop on Very Large Corpora, June 26-30, Massachusetts Institute of Technology, Cambridge, Massachusetts, United States*, pages 82–94.

Ratnaparkhi, A. (1996). A Maximum Entropy Model for Part-Of-Speech Tagging. In *In Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 1996), May 17-18, Philadelphia, Pennsylvania, United States*, pages 133–142.

Ratnaparkhi, A. (1998). *Maximum Entropy Models for Natural Language Ambiguity Resolution*. PhD thesis, University of Pennsylvania, United States.

Roark, B., Saraclar, M., Collins, M., and Johnson, M. (2004). Discriminative Language Modeling with Conditional Random Fields and the Perceptron Algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004), July 21-26, Barcelona, Spain*.

Rosen, B. E. (1996). Ensemble Learning using Decorrelated Neural Networks. *Connection Science*, 8(3–4):373–384.

Schmid, H. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing, September 14-16, Manchester, United Kingdom*, pages 44–49.

Sekine, S., Grishman, R., and Shinnou, H. (1998). A Decision Tree Method for Finding and Classifying Names in Japanese Texts. In *Proceedings of the 6th Workshop on Very Large Corpora, August 15-16, University of Montreal, Montreal, Quebec, Canada*.

Sekine, S. and Isahara, H. (1999). IREX Project Overview. In *Proceedings of the IREX Workshop, September 1-3, Tokyo, Japan*.

Sha, F. and Pereira, F. (2003). Shallow Parsing with Conditional Random Fields. In *Proceedings of the Human Technology Conference - North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003), May 27 - June 1, Edmonton, Canada*.

Sminchisescu, C., Kanaujia, A., Li, Z., and Metaxas, D. (2005). Conditional Models for Contextual Human Motion Recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'05), October 15-21, Beijing, China*, volume 2, pages 1808–1815.

Smith, A., Cohn, T., and Osborne, M. (2005). Logarithmic Opinion Pools for Conditional Random Fields. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005), June 25-30, Ann Arbor, Michigan, United States*.

Smith, A. and Osborne, M. (2005). Regularisation Techniques for Conditional Random Fields: Parameterised Versus Parameter-Free. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP 2005), October 11-13, Jeju Island, South Korea*, pages 896–907.

Smith, A. and Osborne, M. (2006). Using Gazetteers in Discriminative Information Extraction. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X), June 8-9, New York, United States*.

Sollich, P. and Kroghtree-s, A. (1995). Learning with Ensembles: How Over-fitting Can Be Useful. In *Advances in Neural Information Processing Systems 8, November 27-30, Denver, Colorado, United States*.

Stolz, W. S., Tannenbaum, P. H., and Carstensen, F. V. (1965). A Stochastic Approach to the Grammatical Coding of English. *Communications of the Association for Computing Machinery*, 8:399–405.

Sutton, C., Sindelar, M., and McCallum, A. (2006). Reducing Weight Undertraining in Structured Discriminative Learning. In *Proceedings of Human Language Technology - North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2006), June 5-7, New York, United States.*

Tang, J., Hong, M., Li, J., and Liang, B. (2006). Tree-Structured Conditional Random Fields for Semantic Annotation. In *Proceedings of 5th International Conference of Semantic Web (ISWC 2006), November 5-9, Athens, Greece.*

Torralba, A., Murphy, K. P., and Freeman, W. T. (2005). Contextual Models for Object Detection using Boosted Random Fields. In *Advances in Neural Information Processing Systems 17, December 13-16, Vancouver, Canada*.

Viterbi, A. J. (1967). Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, IT 13:260–269.

Wallach, H. (2002). Efficient Training of Conditional Random Fields. Master's thesis, School of Informatics, University of Edinburgh, United Kingdom.

Wang, S. B., Quattoni, A., Morency, L.-P., and Demirdjian, D. (2006). Hidden Conditional Random Fields for Gesture Recognition. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), June 17-22, New York, United States*, pages 1521–1527.

Wang, Y. and Ji, Q. (2005). A Dynamic Conditional Random Field Model for Object Segmentation in Image Sequences. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), June 20-25, San Diego, California, United States*, volume 1, pages 264–270.

Wellner, B., McCallum, A., Peng, F., and Hay, M. (2004). An Integrated, Conditional Model of Information Extraction and Coreference with Application to Citation Matching. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI-2004), July 7-11, Banff, Canada*.

Williams, P. (1995). Bayesian Regularisation and Pruning using a Laplace Prior. *Neural Computation*, 7(1):117–143.

Yeh, A. (2000). More Accurate Tests for the Statistical Significance of Result Differences. In *Proceedings of the 18th Conference on Computational Linguistics (COLING 2000), July 31 - August 4, Saarbrücken, Germany*, pages 947–953.

Zenobi, G. and Cunningham, P. (2001). Using Diversity in Preparing Ensembles of Classifiers Based on Different Feature Subsets to Minimize Generalization Error. *Lecture Notes in Computer Science*, 2167:576–587.